مبادئ وتقنيات علم البيانات

مقدمة الكتاب

هذا الكتاب يخص مادة داتا 100، مبادئ وتقنيات علم البيانات في جامعة كاليفورنيا - بيركلي.

كتبه: سام لاو، جوي غونزاليس و ديب نولان.

ترجم للعربية بواسطة علي العوهلي.

استخدم المترجم أسلوب الترجمة التحريرية لإيصال أفكار الكاتب بشكل صحيح ودقيق للقارئ العربي. سيتم كتابة المصطلحات الإنجليزية بجانب العربية إذا استوجب الأمر للتوضيح، العمليات الحسابية والرياضية ستكون باللغة الإنجليزية. سيضيف المترجم روابط ومراجع لبعض العناوين التي لم يشرحها الكاتب بشكل مفصل وتحتاج مزيد من القراءة سواء في سطر تعليق أو عبر إضافة إشارة (📝) لكل رابط.

مثال: "توجد أنواع عديدة من دوال التنشيط Activation Functions مثل Tanh و ReLu. 📝 📝"
كل 📝 هنا لرابط مختلف.

> مثال لسطر تعليق للمترجم

للحصول على أفضل النتائج للأكواد البرمجية في هذا الكتاب ينصح بإستخدام Jupyter Notebook وهي أداة تسمح لك بإنشاء ومشاركة صفحات يمكن من خلالها كتابة أكواد برمجية، عمليات حسابية ورسوم بيانية، ينصح بمشاهدة هذا الفيديو لمعلومات أكثر عن جوبتر.

تم أخذ الموافقة على ترجمة هذا الكتاب من سام لاو.

داتا 100 هي مادة متقدمة، وتستمر على طول السنة التعليمية وتأتي بعد داتا 8، مبادئ علم البيانات.

عن الكتاب

من المتوقع أن يكون القارئ على دراية بالمعلومات المقدمة في مادة داتا 8 أو ما يعادلها. بالأخص نتوقع أن يكون القارئ على علم بالمواضيع التالية (روابط للصفحات في مادة داتا 8 متوفرة بجانب كُل عنوان):

- معالجة البيانات المجدولة: الاختيار، الفلترة، التجميع (رابط).
- مفاهيم الاحتمالات الأساسية (رابط).
- العينات، التوزيعات التجريبية في الإحصاء (رابط).
- اختبار الفرضيات باستخدام العينات العشوائية (Bootstrap) (رابط). 📝
- الانحدار في المربعات اَلدُّنيَا واستنباطه (رابط). 📝
- التصنيف (رابط).

بالإضافة إلى ذلك، نتوقع أن القارئ أخذ مادة في البرمجة بلغة بايثون، مثلاً CS61A أو ما يشابهها. لن نقوم بشرح كود بايثون إلا في حالات خاصة.

أخيراً، نتوقع أيضاً أن القارئ لديه معلومات أساسية عن المشتقات الجزئية 📝 📝 ، التدرج في حساب المشتقات 📝 📝، جبر المتجهات 📝، و جبر المصفوفات 📝 📝.

الرموز والإشارات

يغطي هذا الكتاب مواضيع مختلفة من عدة مجالات، وللأسف أن بعض هذه المجالات نستخدم نفس الرموز لوصف موضوع ما. ولإبعاد الشبهة، قمنا بإنشاء رموز مخصصة مختلفة قليلاً عما قد تكون تعرفه مُسبقاً.

مغلمة المجتمع الإحصائي Population رُمز لها ب θ^*. قيمة مغلمة النموذج التي تقلل من دالة الخسارة رُمز لها ب $\hat{\theta}$. عادةً، نحاول إيجاد $\hat{\theta} \approx \theta^*$.
نقوم باستخدام الرمز θ دون أي علامة، للإشارة إلى مغلمة النموذج التي لا تقلل من دالة الخسارة. على سبيل المثال، يمكننا أن نفترض قيمة $\theta = 16$ لحساب نتيجة الخسارة للنموذج في تلك القيمة θ. عند استخدام النزول الاشتقاقي Gradient Descent للتقليل من دالة الخسارة، سنستخدم $\theta^{(t)}$ للتعبير عن قيم θ.

سنستخدم دائماً حروف إنجليزية صغيرة وغامقة للتعبير عن المُتجهات Vectors. مثلاً، نستخدم التالي للتعبير عن مُتجه لمغلمات المجتمع الإحصائي $\boldsymbol{\theta}^* = [\theta_1^*, \theta_2^*, \ldots, \theta_n^*]$ ومصفوفة مغلمات النموذج المُدَرب تكون $\hat{\boldsymbol{\theta}} = [\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n]$. وسنستخدم دائماً الحروف الإنجليزية الكبيرة الغامقة للتعبير عن المصفوفات. مثلاً، عادةً ما نرمز لمصفوفة البيانات بالرمز $\boldsymbol{X}$. أيضًا سنستخدم دائمًا الحروف الكبيرة غير الغامقة لوصف المتغيرات العشوائية، كـ X أو Y.

عند الحديث عن التمهيد bootstrap، سنستخدم θ^* للتعبير عن مغلمة المجتمع الإحصائي، و $\hat{\theta}$ للتعبير عن إحصائية اختبار العينة، و $\tilde{\theta}$ للتعبير عن إحصائية اختبار تم استخدام الbootstrap فيها.

دورة حياة علم البيانات

مقدمة

في علم البيانات، نستخدم بيانات عديدة ومتنوعة لاتخاذ قراراتنا. في هذا الكتاب سنشرح مبادئ وتقنيات علم البيانات من الجانب الحسابي والتفكير الاستدلالي. وتشمل الخطوات التالية:

- تشكيل السؤال أو المشكلة.
- إيجاد وتنظيف البيانات.
- التحليل الاستكشافي للبيانات.
- استخدام التوقع والاستدلال لإيجاد النتائج.

ومن المتوقع أن تظهر مزيد من الأسئلة والمشاكل بعد آخر خطوة، في ذلك الوقت يمكننا إعادة الخطوات مرة أخرى لاكتشاف أي خصائص جديدة في مشكلتنا. هذا التكرار الإيجابي في عملنا يسمى **دورة حياة علم البيانات**.

إذا كانت دورة حياة علم البيانات سهلة، لما احتجنا كتباً لشرحها. لحسن حظنا، كل خطوة لديها عدد مختلف من التحديات التي تكشف لنا أفكار جديدة تكون هي أساساً لاتخاذ قرارات مدروسة باستخدام البيانات.

طلاب داتا 100

دورة حياة علم البيانات تتكون من الخطوات التالية:

1- **تشكيل السؤال أو المشكلة**:

- ما الذي نريد معرفته، أو ما هي المشكلة التي نريد حلها؟
- ما هي الفرضيات؟
- ما هي مقاييس نجاحنا؟

2- **إيجاد وتنظيف البيانات**:

- ما هي البيانات المتوفرة لدينا وما هي التي نبحث عنها؟
- كيف سنتمكن من جمع المزيد من البيانات؟
- كيف نرتب البيانات لنبدأ التحليل؟

3- **التحليل الاستكشافي للبيانات**:

- هل لدينا بيانات ذات علاقة بمشكلتنا؟
- هل تحتوي البيانات على تحيزات، بيانات شاذة، أو مشاكل أخرى؟
- كيف نحول البيانات لتساعدنا على القيام بتحليل فعال؟

4- **التوقع والاستدلال**:

- ماذا تخبرنا البيانات؟
- هل أجابت على السؤال أو حلت المشكلة؟
- ما مدى قوة نتائجنا؟

سنقوم الآن بتجربة هذه الخطوات على قاعدة بيانات الأسماء الأولى لطلاب داتا 100 من الفصول السابقة. في هذا الفصل، قمنا بالمرور بشكل سريع على الخطوات لإعطاء القارئ معلومات عن الدورة الكاملة. في فصول لاحقة، سنتحدث بشكل مفصل ونشرح كل خطوة.

تشكيل السؤال أو المشكلة

نريد أن نعرف ما إذا كانت الأسماء الأولى للطلاب تقدم لنا معلومات إضافية عنهم. رغم أن السؤال يبدو غامضاً نوعاً ما، لكنه كافياً لجعلنا نعمل على البيانات المتوفرة لدينا ويمكننا التعديل في السؤال أثناء عملنا لنجعله أكثر دقة.

إيجاد وتنظيف البيانات

لنبدأ بأخذ نظرة سريعة عن البيانات المتوفرة لدينا، البيانات هي قائمة لأسماء الطلاب الأولى للذين سبق أن درسوا مادة داتا 100. **لا تقلق إن لم تفهم الكود البرمجي؛ سنشرحه ونشرح المكتبات المستخدمة لاحقاً**. حالياً، ركز على الخطوات والرسوم البيانية:

لتحميل بيانات اسماء الطلاب اضغط هنا.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

students = pd.read_csv('roster.csv')
students
```

	Role	Name
0	Student	Keeley
1	Student	John
2	Student	BRYAN
...	...	...
276	Waitlist Student	Ernesto
277	Waitlist Student	Athan
278	Waitlist Student	Michael

279 rows × 2 columns

يمكن أن نلاحظ بشكل سريع وجود بعض المشاكل في بياناتنا. مثلاً، أحد الطلاب كُتب اسمه بالأحرف الكبيرة بشكل كامل BRYAN، بالإضافة إلى أن معنى العمود Role لا يبدو واضحاً. نلاحظ أيضاً أن الجدول يحتوي على عامودين و 279 سطراً.

في هذه المادة، سنتعلم كيفية اكتشاف الأخطاء في بياناتنا وتصحيحها. الاختلاف في الحروف الكبيرة في الاسم Bryan سيجعل البرنامج يتوقع أن BRYAN يختلف عن Bryan ولكن في الحقيقة هما نفس الشخص. لذا سنحول جميع الأسماء إلى حروف صغيرة:

```
students['Name'] = students['Name'].str.lower()
students
```

	Role	Name
0	Student	keeley
1	Student	john
2	Student	bryan
...	...	...
276	Waitlist Student	ernesto
277	Waitlist Student	athan
278	Waitlist Student	michael

279 rows × 2 columns

الآن، وبما أن البيانات لدينا بدأت تظهر بشكل مقبول، يمكننا الانتقال للخطوة التالية.

التحليل الاستكشافي للبيانات

التحليل الاستكشافي للبيانات أو Exploratory Data Analysis واختصارها EDA يطلق على الخطوات التي نتبعها لمعرفة صفات البيانات لعمل تحليلات لها لاحقاً. لنستعرض بيانات الطلاب:

```
students
```

	Role	Name
0	Student	keeley
1	Student	john
2	Student	bryan
...	...	...
276	Waitlist Student	ernesto
277	Waitlist Student	athan
278	Waitlist Student	michael

279 rows × 2 columns

الآن لدينا بعض الأسئلة، كم عدد الطلاب؟ ماذا يعني عمود Role؟ نقوم بخطوة التحليل الاستكشافي للبيانات للإجابة على مثل هذه الأسئلة.

كم عدد الطلاب؟

```
print("There are", len(students), "students on the roster.")
```

```
There are 279 students on the roster.
```

عدد الطلاب لدينا هو 279 طالباً. السؤال التالي دائماً يكون: هل تحتوي البيانات على كامل الطلاب؟ في حالتنا، الجدول يحتوي على جميع الطلاب الذين درسوا مادة داتا 100 في فصلٍ دراسيٍّ واحد.

ماذا يعني عمود Role؟

لنستكشف البيانات التي في هذا العمود لنعرف معناه:

```
students['Role'].value_counts().to_frame()
```

	Role
Student	237
Waitlist Student	42

يمكن أن نرى في الجدول السابق أن البيانات لا تحتوي فقط على الطلاب الذين درسوا المادة Student، بل أيضاً على الطلاب الموجودين في قائمة الانتظار Waitlist Student. إذاً، العمود Role يخبرنا إذا كان الطالب التحق بالمادة أم لا.

ماذا عن عمود Name؟ كيف يمكننا استكشافه؟

في هذه المادة سنتعامل مع عدد كبير من أنواع البيانات. الرقمية، النوعية والنصية. كل نوع له أساليبه وأدواته الخاصة للاستكشاف.

طريقة سريعة لفهم عمود الأسماء Name هي بمعرفة عدد الأحرف في كل اسم:

```
sns.distplot(students['Name'].str.len(),
             rug=True,
             bins=np.arange(12),
             axlabel="Number of Characters")
plt.xlim(0, 12)
plt.xticks(np.arange(12))
plt.ylabel('Proportion per character');
```

الرسم البياني السابق يخبرنا أن أكثر الأسماء يبلغ طولها بين 4 إلى 8 أحرف. هذا يساعدنا على معرفة ما إذا كانت بياناتنا معقولة أم لا. إذا كان هناك الكثير من الأسماء ذات حرف واحد، فيكون ذلك سبب مناسب لإعادة استكشاف البيانات.

رغم أن البيانات تبدو واضحة وبسيطة، سنعرف لاحقاً كيف أن الاسم الأول فقط قد يخبرنا الكثير عن مجموعة الطلاب لدينا.

ماذا بداخل عمود الاسم؟

حتى الآن، وجهنا سؤال عام: "هل يخبرنا الاسم الأول للطالب أي شيء عن المادة؟"

قمنا بتنظيف البيانات بتحويلها جميعها لأحرف صغيرة. أثناء التحليل الاستكشافي للبيانات لاحظنا أن لدينا حوالي 270 طالباً منهم من درس المادة ومنهم من على قائمة الانتظار. وأكثر الأسماء بين 4 إلى 8 أحرف.

ماذا يمكننا معرفته عن طلاب المادة من أسمائهم؟ لنأخذ اسماً واحداً منها:

```
students['Name'][5]
```

```
'jerry'
```

من هذا الاسم، يمكننا القول أن صاحب الاسم ذكر. ويمكننا أيضاً توقع عمر الطالب. على سبيل المثال، إذا عرفنا أن اسم Jerry مشهور من بين أسماء الأطفال الذين ولدو في عام 1998، يمكننا التوقع إن عمر الطالب في العشرينيات.

التفكير بهذه الطريقة أوصلنا إلى سؤالين:

- هل تخبرنا أسماء الطلاب عن توزيع الذكور والإناث؟
- هل تخبرنا أسماء الطلاب عن توزيع الأعمار؟

للإجابة على هذه الأسئلة، سنحتاج بيانات تربط بين الأسماء مع الجنس والسنوات. مؤسسة الضمان الاجتماعي الأمريكية لديها مثل هذه البيانات ومتوفرة على الإنترنت على الرابط.

سنبدأ أولاً بتحميل البيانات من الموقع ثم نقلها إلى بايثون. مرة أخرى، **لا تقلق إذا لم تفهم الكود البرمجي في هذا الفصل، فقط ركز على فهم الخطوات بشكل عام**:

```
import urllib.request
import os.path

data_url = "https://www.ssa.gov/oact/babynames/names.zip"
local_filename = "babynames.zip"
if not os.path.exists(local_filename): # اذا توفرت البيانات، لا تحملها مرة أخرى
    with urllib.request.urlopen(data_url) as resp, open(local_filename, 'wb') as f:
        f.write(resp.read())

import zipfile
babynames = []
with zipfile.ZipFile(local_filename, "r") as zf:
    data_files = [f for f in zf.filelist if f.filename[-3:] == "txt"]
    def extract_year_from_filename(fn):
        return int(fn[3:7])
    for f in data_files:
        year = extract_year_from_filename(f.filename)
        with zf.open(f) as fp:
            df = pd.read_csv(fp, names=["Name", "Sex", "Count"])
            df["Year"] = year
            babynames.append(df)
babynames = pd.concat(babynames)
babynames
```

	Name	Sex	Count	Year
0	Mary	F	9217	1884
1	Anna	F	3860	1884
2	Emma	F	2587	1884
		...	...	...
2081	Verna	M	5	1883
2082	Winnie	M	5	1883
2083	Winthrop	M	5	1883

1891894 rows × 4 columns

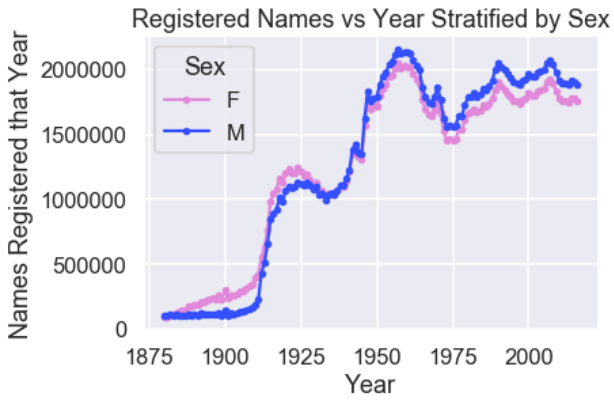
البيانات تحتوي على الأسماء، جنس الطفل، عدد الأطفال بهذا الاسم وسنة ميلاد كل طفل. للتأكيد، لنقرأ ما كتبه مكتب الضمان الاجتماعي في شرحهم للبيانات على الرابط.

> جميع الأسماء أتت من بطاقات التقديم للضمان الاجتماعي لجميع الولادات التي كانت في الولايات المتحدة بعد عام 1879. **ملاحظة**: الكثير ممن ولدوا قبل 1937 لم يقوموا بالتقديم للحصول على البطاقة، لذلك أسماؤهم ليست ضمن البيانات. الآخرين الذين قاموا بتقديم طلباتهم، سجلاتنا لا تُظهر أماكن ولاداتهم، لذلك أسماؤهم أيضاً لم تضاف إلى البيانات. هذه عينة كاملة من البيانات لدينا حتى تاريخ مارس 2017.

نبدأ أولاً بعرض عدد المواليد الذكور والإناث كل سنة:

```
pivot_year_name_count = pd.pivot_table(
    babynames, index='Year', columns='Sex',
    values='Count', aggfunc=np.sum)
```

```
pink_blue = ["#E188DB", "#334FFF"]
with sns.color_palette(sns.color_palette(pink_blue)):
    pivot_year_name_count.plot(marker=".")
    plt.title("Registered Names vs Year Stratified by Sex")
    plt.ylabel('Names Registered that Year')
```



الزيادة المفاجئة لعدد المواليد في عام 1920 تبدو مشبوهة، ولكن في الاقتباس السابق تم توضيح السبب:

> **ملاحظة**: الكثير ممن ولدوا قبل 1937 لم يقوموا بالتقديم للحصول على البطاقة، لذلك أسماؤهم ليست ضمن البيانات. الآخرين الذين قاموا بتقديم طلباتهم، سجلاتنا لا تُظهر أماكن ولاداتهم، لذلك أسماؤهم أيضاً لم تضاف إلى البيانات.

يمكن ملاحظة فترة الإنجاب المتزايدة أو ما تسمى ب Baby boomers والتي كانت في الفترة بين 1946 حتى 1964، لقراءة المزيد عن هذا الموضوع قم بزيارة الرابط.

معرفة الجنس من الاسم

لنستخدم بيانات الأطفال السابقة لمعرفة عدد الذكور والإناث. كما فعلنا سابقاً، نبدأ أولاً بتصغير جميع أحرف الأسماء في بيانات الأطفال:

```
babynames['Name'] = babynames['Name'].str.lower()
babynames
```

	Name	Sex	Count	Year
0	mary	F	9217	1884
1	anna	F	3860	1884
2	emma	F	2587	1884
		...	...	...
2081	verna	M	5	1883
2082	winnie	M	5	1883
2083	winthrop	M	5	1883

2084 rows × 4 columns

ثم نجمع عدد المواليد لكل اسم ونوع المولود:

```
sex_counts = pd.pivot_table(babynames, index='Name', columns='Sex',
                            values='Count', aggfunc='sum',
                            fill_value=0., margins=True)
sex_counts
```

Sex	F	M	All
Name			
aaban	0	96	96
aabha	35	0	35
aabid	0	10	10
	...	...	...
zyyon	0	6	6

Sex	F	M	All
zzyzx	0	5	5
All	170639571	173894326	344533897

96175 rows × 3 columns

لتحديد ما إذا كان الاسم أكثر شيوعاً للأطفال الذكور أم الإناث، يمكننا حساب نسبة تكرار الاسم لدى أحد الجنسين:

```
prop_female = sex_counts['F'] / sex_counts['All']
sex_counts['prop_female'] = prop_female
sex_counts
```

Sex	F	M	All	prop_female
Name				
aaban	0	96	96	0.0
aabha	35	0	35	1.0
aabid	0	10	10	0.0
		...	...	...
zyyon	0	6	6	0.0
zzyzx	0	5	5	0.0
All	170639571	173894326	344533897	0.5

96175 rows × 4 columns

يمكننا تعريف دالة لتبحث لنا عمّا إذا كان الاسم ذكراً أو أنثى باستخدام النسبة السابقة:

```
def sex_from_name(name):
    if name in sex_counts.index:
        prop = sex_counts.loc[name, 'prop_female']
        return 'F' if prop > 0.5 else 'M'
    else:
        return 'Name not in dataset'

students['sex'] = students['Name'].apply(sex_from_name)
sex_from_name('sam')
```

```
'M'
```

باستخدام الكود البرمجي السابق يمكنك تجربة أي اسم ومعرفة ما إذا كانت نسبة تسميته كذكر أعلى من نسبة تسميته كأنثى.

الآن لنعود إلى بيانات الطلاب ونضيف عليها ما إذا كان الطالب ذكراً أو أنثى:

	Role	Name	Sex
0	Student	keeley	F
1	Student	john	M
2	Student	bryan	M
	...	...	...
276	Waitlist Student	ernesto	M
277	Waitlist Student	athan	M
278	Waitlist Student	michael	M

279 rows × 4 columns

الآن يمكننا بسهولة معرفة عدد الذكور والإناث بين طلابنا:

```
students['sex'].value_counts()
```

```
M                    144
F                     92
Name not in dataset   43
Name: Sex, dtype: int64
```

إيجاد العمر من الاسم

باستخدام نفس الطريقة السابقة يمكننا إيجاد توزيع العمر في فصلنا، بربط كل اسم مع متوسط السنوات الذي تكرر فيها:

```
def avg_year(group):
    return np.average(group['Year'], weights=group['Count'])

avg_years = (
    babynames
    .groupby('Name')
    .apply(avg_year)
    .rename('avg_year')
    .to_frame()
)
avg_years
```

	avg_year
Name	
aaban	2012.57
aabha	2013.71
aabid	2009.50
...	...
zyyanna	2010.00
zyyon	2014.00
zzyzx	2010.00

96174 rows × 1 columns

بنفس الطريقة السابقة، يمكن أن نبحث عن أي اسم ومعرفة متوسط سنة الميلاد:

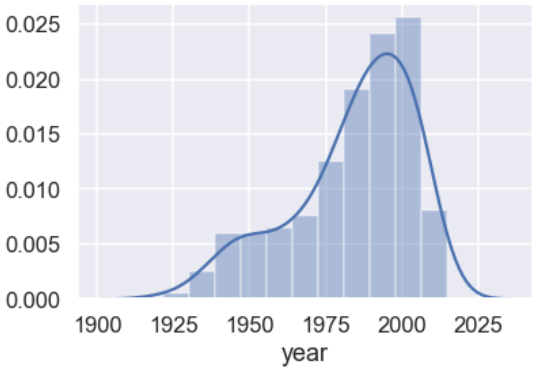
```
def year_from_name(name):
    return (avg_years.loc[name, 'avg_year']
            if name in avg_years.index
            else None)

students['year'] = students['Name'].apply(year_from_name)
students
```

	Role	Name	Sex	Year
0	Student	keeley	F	1998.15
1	Student	john	M	1951.08
2	Student	bryan	M	1983.57
		...	...	...
276	Waitlist Student	ernesto	M	1981.44
277	Waitlist Student	athan	M	2004.40
278	Waitlist Student	michael	M	1971.18

279 rows × 4 columns

الآن، يمكننا بسهولة عرض توزيع السنوات بين الطلاب:

```
sns.distplot(students['year'].dropna());
```



ولعرض متوسط عمود السنة نقوم بالآتي:

```
students['year'].mean()
```

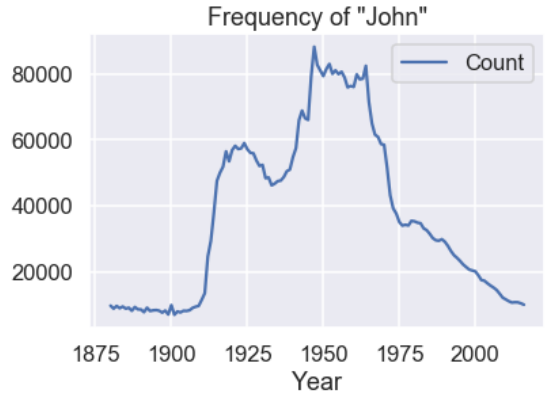
```
1983.846741800525
```

متوسط الأعمار لدينا هو 35 سنه (2018-1983=35)، تقريباً أكثر بمرتين من العمر المتوقع للطلاب الجامعيين. لماذا تظهر لنا الأعمار مرتفعة بهذا الشكل؟

كعالم بيانات، قد نصل لنتائج لا نتفق معها أو عكس توقعاتنا. التحدي الدائم الذي يواجهنا هو معرفة ما إذا كانت النتائج التي فاجأتنا سببها خطأ في إحدى خطواتنا أو خطأ حقيقي في البيانات. بما أنه لا يوجد هناك طريقة سهلة لضمان نتائج دقيقة، يجب أن يكون لدى عالم البيانات مبادئ وقواعد للتقليل من إيجاد نتائج خاطئة.

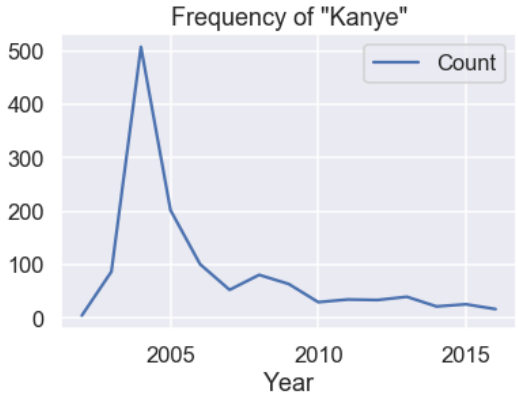
في حالتنا، التفسير الوحيد للنتيجة غير المتوقعة التي ظهرت لنا هو أن الأسماء الأكثر شيوعاً تستخدم منذ سنوات عديدة. مثلاً الاسم John يعتبر من الأسماء الأكثر شيوعاً عبر التاريخ بناءً على البيانات التي حصلنا عليها. يمكننا تأكيد ذلك بعرض رسم بياني لعدد الأطفال الذين تم تسميتهم John كل سنة:

```
names = babynames.set_index('Name').sort_values('Year')
john = names.loc['john']
john[john['Sex'] == 'M'].plot('Year', 'Count')
plt.title('Frequency of "John"');
```



يبدو لنا أن متوسط السنة لا يعطي توقع دقيق لعمر الشخص. ولكن في بعض الحالات، الاسم الأول للشخص يساعدنا على ذلك، على سبيل المثال عند التجربة على اسم Kanye تظهر لنا النتيجة التالية:

```
names = babynames.set_index('Name').sort_values('Year')
kanye = names.loc['kanye']
kanye[kanye['Sex'] == 'M'].plot('Year', 'Count')
plt.title('Frequency of "Kanye"');
```



الملخص

في هذا الفصل، قمنا بالمرور بشكل سريع على دورة حياة علم البيانات: تشكيل السؤال أو المشكلة، إيجاد وتنظيف البيانات، التحليل الاستكشافي للبيانات، التوقع والاستدلال. سنشرح بشكل مفصل كل خطوة في الفصول القادمة.

النصف الأول من الكتاب (الفصل 1 حتى 9) سيغطي الثلاث خطوات الأولى من دورة حياة علم البيانات ويركز بشكل كبير على طريقة الحساب وإيجاد النتائج. النصف الثاني (الفصل 10 حتى 18) يستخدم التفكير الحسابي والإحصائي لتغطية مواضيع بناء النماذج، الاستدلال والتوقع.

بشكل عام، يسعى هذا الكتاب لتعريف القارئ على مبادئ وتقنيات علم البيانات.

تصميم البيانات

مقدمة

البيانات هي عصب علم البيانات. لذا من الصعب البدء بتحليل البيانات دون معرفة كيف تم جمعها.

في هذا الفصل سنتحدث عن **تصميم البيانات**، خطوة جمع وتكوين البيانات. الكثير من علماء البيانات استنتجوا نتائج مبكرة غير صحيحه لأنهم لم يتمكنوا من فهم بياناتهم. سنقوم باستعراض أمثله لتأكيد أهمية هذه النقطة، وأهمية استخدام الاحتمالات في علم البيانات لإيجاد العينات .

ديوي يهزم ترومان

في صراع الرئاسة الأمريكية عام 1948، كان عمدة نيويورك "توماس ديوي" منافساً لـ "هاري ترومان". كالعادة، أجرت عدة وكالات مختصه بالاستطلاع والتصويت بعض الاستطلاعات للتنبؤ بالفائز في السباق الرئاسي.

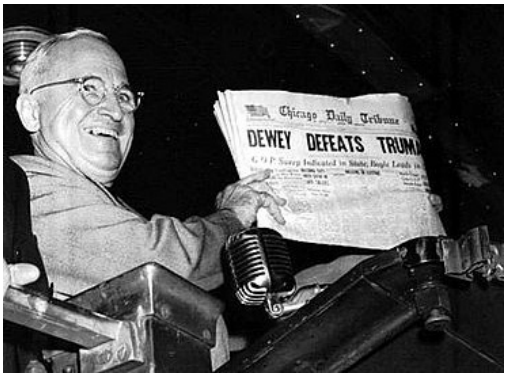
1936: كارثة الاستطلاع السابق

في عام 1936، أي ثلاث سباقات انتخابية قبل 1948، توقعت مجلة The Literary Digest خسارة "فرانكلين روزفلت" وفوز ساحق لمنافسه "آلف لاندن". وكان تبرير المجلة أنها قامت بأخذ عينه تصويت لأكثر من مليونين شخص اعتماداً على أرقام الهواتف ومعلومات السيارات المسجلة. وكما يبدو واضحاً، هذه العينة تحتوي على تحيز كبير، لأن في تلك الفترة كان أغلب مالكي الهواتف والسيارات أغنى ممّن لا يملكونها. في هذه الحالة، التحيز كان كبيراً حيث جعل المجلة تتوقع أن "روزفلت" سيحصل على 43% من إجمالي الأصوات، ولكنه في النهاية حصل على 61%، بفارق حوالي 20% عن توقع المجلة ويعتبر من أكبر الأخطاء التي حدثت في استطلاع عام. بعد ذلك بفتره قصيرة، أغلقت المجلة ولم تعد تعمل. 📝

1948: تصويت غالوب

رغبة في التعلم من الأخطاء السابقة، قامت "مؤسسة غالوب" باستخدام طريقة سميت ب "عينة الحصة Quota Sampling" لتوقع نتيجة انتخابات الرئاسة الأمريكية لعام 1948. في فترة جمع العينات، كل مُحاوِر استطلع رأي مجموعة أشخاص من كل فئة ديموغرافية. مثلاً، كل محاور يجب أن يستطلع رأي الذكور والإناث، بمختلف الأعمار، الأعراق، ومستوى الدخل، ليطابق التراكيب السكانية في تعدادت الولايات المتحدة. تم ذلك للتأكد أن التصويت لن يترك أي فئة من مجموع المصوتين.

باستخدام هذه الطريقة، توقعت "غالوب" أن "توماس ديوي" سيكسب أصواتاً أكثر من "هاري ترومان" بنسبة 5%. الفارق كان ملحوظاً ليجعل صحيفة Chicago Tribune تطبع العنوان بشكل عريض "ديوي يهزم ترومان":



كما نعرف الآن، "ترومان" هو من انتصر بسباق الرئاسة. بالأصح، فاز بفارق 5% عن "ديوي"! ما الخطأ الذي حدث في تصويت "مؤسسة غالوب"؟

المشكلة في العينة الحصصية

رغم أن الطريقة تساعد منظمي الاستطلاعات على تقليل الانحياز في جمع البيانات، لكنها أظهرت انحيازاً من جهة أخرى. حيث أخبرت المؤسسة محاوريها أن بإمكانهم مقابلة أياً كان طالما أنهم ينجزون الحصة المخصصة لهم. هذه إحدى التفسيرات عن سبب حصول المحاورين على عدد غير متناسب من الجمهوريين. في ذلك الوقت، يعتبر الجمهوريون متوسطي الثراء وغالباً ما يعيشون في أحياء راقية، مما يجعلهم أسهل لإجراء المقابلة. يدعم هذا التفسير حقيقة أن المؤسسة توقعت نسبة 2-6% أكثر لأصوات الجمهوريين مقارنة بالأرقام الحقيقية للانتخابات الثلاث السابقة.

هذه الأمثلة تظهر أهمية فهم الانحياز أثناء خطوة جمع البيانات. Literary Digest و "مؤسسة غالوب" كلاهما قامتا بنفس الخطأ، اعتقاداً أن طريقتهم غير منحازة بينما في الحقيقة كلا الطريقتين كانتا تعتمدان على أحكام المحاورين.

نحن الآن نعتمد على **العينات المحتملة**، إحدى طرق جمع العينات والتي تخصص احتمالات معينة لظهور أي عينة، للتقليل من الانحياز في جمع بياناتنا.

البيانات الضخمة؟

في عصر البيانات الضخمة، نحاول التقليل من الانحياز بجمع المزيد من البيانات. في النهاية، نحن نعلم أنها ستعطينا احتمالات مناسبة؛ أليست كل العينات الكبيرة تعطينا احتمالات مناسبة أياً كانت طريقة جمع العينة؟

سنعود لهذا السؤال بعد مناقشة طرق جمع العينات المحتملة لمقارنة كلا الطريقتين.

نظرة عامة على الاحتمالات

الكثير من المبادئ الأساسية في علم البيانات، يشمل ذلك تصميم البيانات، تعتمد على قوانين معينة. قانون الاحتمالات Probability يسمح لنا بتحديد احتمالية حدوث شيء ما. في هذا الفصل سنستعرض مراجعة سريعة عن الاحتمالات وأهميتها.

لنفرض أن لدينا عملتين معدنيتين، بجهة توجد صورة H (Head) وفي الجهة الأخرى كتابة T (Tails). إذا أردنا رمي العملتين ومشاهدة النتيجة نسمي ذلك **بالتجربة Experiment**. و **فضاء النتائج Outcome Space** Ω يشمل جميع النتائج المحتملة لهذه التجربة. فضاء نتائج تجربتنا هو:
$\Omega = \{HH, HT, TH, TT\}$.

الحدث Event هي أحد النتائج من فضاء النتائج. على سبيل المثال، الحصول على H واحد و T واحد هو أحد الأحداث من تجربتنا. هذا الحدث يتكون في النتائج $\{HT, TH\}$. نستخدم الرمز $P(\text{event})$ للتعبير عن احتمالية الحدث، في مثالنا نرمز لها ب:
$P(\text{One Head and One tails})$ (احتمالية ظهور صورة مرة وكتابة مرة).

احتمالية الأحداث

في هذا الكتاب، سنتعامل مع مشاكل لديها فضاءات نتائج تكون فيها احتمالية حدوث أي من النتائج متساوي. هذه الفضاءات لديها معادلة حسابية بسيطة لحساب الاحتمالية. المعادلة هي: عدد النتائج المحتملة للحدث قسمة عدد جميع الأحداث في فضاء النتائج:

$$P(event) = \frac{\#\text{ of outcomes in event}}{\#\text{ of all possible outcomes}}$$ = نسبة احتمالية الحدث

في مثال رمي العملتين المعدنيتين، احتمالية حدوث جميع الأحداث $\{HH, HT, TH, TT\}$ متساوية. لحساب $P(\text{One Head and One tails})$، نجد أن احتمال حدوث الحدث هي 2 من مجموع الأحداث 4:

$$P(\text{One Head and One tails}) = \frac{2}{4} = \frac{1}{2}$$

يوجد لدينا ثلاث أساسيات مسلّم بها في حساب الاحتمالات:

- احتمالية حدوث أي حدث هي عدد حقيقي بين 0 و 1: $0 \leq P(\text{event}) \leq 1$.

- مجموع احتمالية جميع الأحداث في فضاء العينة هو 1: $P(\Omega) = 1$.
- الثالثة تتضمن الأحداث التي لا تحدث مع بعضها Mutually Exclusive مثل:
  - A أو B ($A \cup B$).
  - A و B ($A \cap B$).
  - B بشرط حدوث A أولاً ($B|A$).

A أو B

عادةً ما نحاول إيجاد احتمالية حدوث A أو B. مثلاً، مؤسسة متخصصة في الاقتراع والتصويت تريد إيجاد احتمالية اختيار شخص بشكل عشوائي بحيث يكون عمره 17 أو 18 سنه. ذكرنا أن الأحداث لا يمكن حدوثها في نفس الوقت Mutually Exclusive. إذا كانت A هي اختيارنا لشخص بعمر 17 سنة، و B هي اختيارنا لشخص بعمر 18 سنة، فإن حدوث A و B معاً مستحيل لأنه لا يمكن للشخص أن يكون عمره 17 و 18 في نفس الوقت.

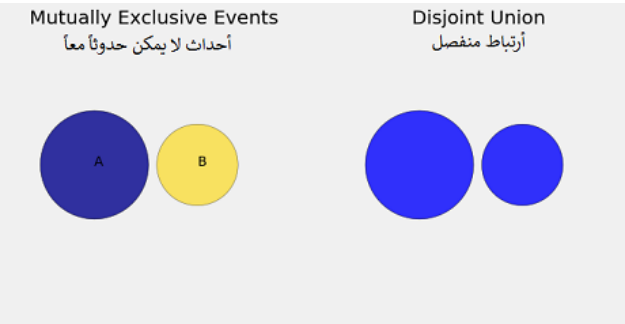
لحساب احتمالية $P(A \cup B)$:

$$P(A \cup B) = P(A) + P(B)$$

التعداد السكاني لعام 2010 في الولايات المتحدة أظهر أن 1.4% من السكان الأمريكيين بعمر 17 سنة، و 1.5% بعمر 18 سنة. لذا إذا اخترنا شخصاً بشكل عشوائي من السكان فإن احتمالية أن عمره 17 سنة هي $P(A) = 0.014$ و احتمالية أن عمره 18 سنة هي $P(B) = 0.015$، ذلك يعني احتمالية اختيارنا لشخص بشكل عشوائي عمره 17 أو 18 سنه هي:

$$P(A \cup B) = P(A) + P(B) = 0.014 + 0.015 = 0.029$$

لتوضيح الفكرة بشكل أكثر لنقم بشرحها على رسم Venn البياني:



عندما يكون الحدثان A و B الذين لا يمكن حدوثهما معاً، فلا توجد نتائج تظهر في A و B معاً. لذا إذا كانت A تحتوي على 5 نتائج و B تحتوي على 4. فإن $A \cup B$ لديها 9 نتائج مُحتملة.

عندما تكون إمكانية ظهور النتائج معاً ممكنة فإنه لا يمكن تطبيق عملية الجمع عليها. لنعود لمثال العملتين المعدنيتين، نجد عند رمي العملة المعدنية أن نتيجة مؤكدة واحدة هي صورة أو نتيجة مؤكدة واحدة هي كتابة.

إذا كانت A هي حدث ظهور صورة مرة واحدة، إذاً الاحتمالية هي $P(A) = \frac{2}{4}$ لأن فضاء A يحتوي على النتائج $\{HT, TH\}$ و مجموع النتائج المحتملة هو 4. وإذا كانت B هي حدث ظهور كتابة مرة واحدة، إذاً احتماله يساوي $P(B) = \frac{2}{4}$ لأن فضاء نتائج B أيضاً يحتوي على $\{HT, TH\}$. ولكن الحدث $A \cup B$ يحتوي على نتيجتين فقط لأن النتيجتين هنا متشابهتان في A و B وهي $P(A \cup B) = \frac{1}{2}$. القيام هنا بجمع النتائج لاحتساب الاحتمالات يوصلنا لنتيجة خاطئة. هناك نتائج تظهر في A و B; إضافة الرقمين لنتائج A و B في هذا المثال فسنكون قد حسبناهما مرتين. في الرسم البياني التالي يظهر الجزء المشترك بين A و B.

لحل هذه المشكلة، نحتاج لطرح احتمالية $P(A \cap B)$. ولحسابها نقوم بالتالي:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

لاحظ أنه عندما تكون احتمالية ظهور الأحداث A و B معاً مستحيلة، فإن $P(A \cap B) = 0$ وبالتالي تظهر المعادلة السابقة.

A و B

نحاول أحياناً إيجاد احتمالية حدوث كلا الحدثين A و B معاً. على سبيل المثال، إذا كان لدينا فصل بثلاث طلاب X، Y و Z. ما هي احتمالية أن نختار عينة واحدة بحجم 2 يكون فيها القيمة الأولى وتليها X؟و دون أن نعيد عملية سحب العينة؟

إحدى الطرق لإيجاد الاحتمالية هي سرد جميع الاحتمالات الممكن حدوثها:

$$\Omega = \{ \text{ XY, XZ, YX, YZ, ZX, ZY } \}$$

بما أننا نرغب بسحب عينه واحدة يكون فيها Y أولاً ويتبعها X فإن النتيجة هي:

$$P(YX) = \frac{1}{6}$$

يمكن تقسيم الاحتمالية إلى قسمين، الأولى هي سحب العنصر Y أولاً، والثانية هي سحب X بعد Y. إذاً احتمالية سحب Y أولاً هي $\frac{1}{3}$ لأن هناك ثلاث احتمالات ممكن أن تُسحب وهي X،Y و Z.

بعد سحب Y، فضاء النتائج لعملية السحب الثانية تحتوي على $\{X, Z\}$، ويعني ذلك أن احتمالية سحب X كعنصر ثاني هي $\frac{1}{2}$. تسمى هذي الاحتمالية ب **الاحتمال الشرطي Conditional Probability**، وهي احتمالية سحب X بعد أن سحبنا Y أولاً. نستخدم الرمز $P(B|A)$ للاحتمالية المشروطة ونعني هنا أن احتمالية حدوث B بشرط حدوث A مُسبقاً.

لنستعرض التالي:

$$\begin{aligned} P(YX) &= \frac{1}{6} \\ &= \frac{1}{3} \cdot \frac{1}{2} \\ &= P(\text{Y first}) * P(\text{X second}|\text{Y first}) \end{aligned}$$

قاعدة الضرب السابقة هي إحدى قواعد الاحتمالات العامة، لأي حدث A و B، احتمالية حدوث كليهما هي:

$$P(AB) = P(A)P(B|A)$$

في بعض الحالات، كلا **الحدثين مستقلين Independent Events** عن بعضهما. احتمالية ظهور B ثانياً لا يتأثر إذا ظهر A أولاً أو لم تظهر. مثلاً، في حجر النرد، ظهور الرقم 6 لا يؤثر شيئاً على احتمالية ظهور الرقم 5 بعده. إذا كانت A و B مستقلين فإن: $P(B|A) = P(B)$. هذه تُبسط لنا قاعدة الضرب:

$$P(AB) = P(A)P(B) \qquad \text{For Independent } A \text{ and } B$$

رغم أن ذلك يسهل لنا عملية الحساب، إلا أن في الواقع وعند مواجهة بيانات حقيقية، الأحداث ليست مستقلة عن بعضها، حتى ولو بدت العلاقة بينهما غير واضحة في البداية. مثلاً، احتمالية الاختيار العشوائي لمواطن أمريكي عمره 90 سنة أو أعلى ليست مستقلة عن كون المواطن ذكر أو أنثى، لأننا أعطينا أن الشخص عمرة 90 سنة، فإن احتمالية أن يكون الشخص أنثى أكثر مرتين من أن يكون ذكراً.

كعلماء بيانات، يجب علينا التحقق من الفرضيات واستقلاليتها بشكل دقيق! سقوط سوق عقار المنازل الأمريكية عام 2008 كان يُمكن تفاديه لو لم تفترض البنوك أن أسعار المنازل تختلف من مدينة لأخرى بشكل مستقل. (المزيد عن ذلك هنا)

للمزيد عن الاحتمالات:

استخدام الاحتمالات لإيجاد العينات

على عكس الطريقة العادية لإيجاد العينات أو السحب العشوائي، استخدام الاحتمالات لإيجاد العينات يمكننا من تحديد بعض الصفات التي يجب توفرها في العينة. توجد طريقتان لإيجاد العينات باستخدام الاحتمالات: العينة العنقودية Cluster Sampling و العينة الطبقية Stratified Sampling.

إذا كان لدينا 6 أشخاص. وأعطينا كل شخص حرفاً مختلفاً من A إلى F:

العينة العشوائية البسيطة

للحصول على عينة عشوائية ذات حجم 2 من مجموعة الأشخاص، يمكننا كتابة حرف كل شخص على ورقة، وضعها في صندوق، خلطها، ثم سحب ورقتين. ببساطة هذه هي طريقة العينة العشوائية البسيطة.

النتائج المحتملة ذات الحجم 2 هي:

$$AB \quad BC \quad CD \quad DE \quad EF$$

$$AC \quad BD \quad CE \quad DF$$

$$AD \quad BE \quad CF$$

$$AE \quad BF$$

$$AF$$

يوجد لدينا 15 عينة محتمل حدوثها ذات الحجم 2 من مجموع الأشخاص الستة. طريقة أخرى لحساب مجموع العينات المحتملة: 📝

$$\binom{6}{2} = \frac{6!}{2!(6-2)!} = \frac{6!}{2!4!} = \frac{6 \times 5 \times 4 \times 3 \times 2 \times 1}{(2 \times 1) \times (4 \times 3 \times 2 \times 1)} = \frac{720}{2 \times 24} = \frac{720}{48} = 15$$

بما أن السحب يتم بطريقة عشوائية وجميع العينات متساوية باحتمالات سحبها، فإن احتمالية سحب كل عينة من الـ15 نتائج المحتملة هي:

$$P(AB) = P(CD) = \ \ldots \ = P(DF) = \frac{1}{15}$$

يمكننا أيضاً توقع احتمالية ظهور عنصر معين في العينة بحجم 2. مثلاً، العنصر A يظهر 5 مرات في القوائم التي عرضناها سابقاً، إذاً:

$$P(A \text{ in sample}) = \frac{5}{15} = \frac{1}{3}$$

ينطبق هذا المثال على جميع العناصر الأخرى:

$$P(A \text{ in sample}) = P(F \text{ in sample}) = \frac{1}{3}$$

توجد طريقة أخرى لحساب احتمالية ظهور عنصر معين. مثلاً لظهور A في العينة، يجب علينا سحبها أولاً أو ثانياً. إذاً:

$$\begin{aligned} P(A \text{ in sample}) &= P(\text{A is first or A is second}) \\ &= P(\text{A is first}) + P(\text{A is second}) \\ &= \frac{1}{6} \cdot \frac{5}{5} + \frac{5}{6} \cdot \frac{1}{5} \\ &= \frac{1}{3} \end{aligned}$$

العينة العنقودية Cluster Sampling

في العينة العنقودية نقوم بتقسيم البيانات إلى مجموعات. ثم نستخدم مفاهيم العينة العشوائية البسيطة لاختيار المجموعات بشكل عشوائي.

مثلاً، إذا كان لدينا 6 أشخاص ثم جمعناهم في مجموعات من شخصين: (A, B) (C, D) (E, F) ليتم تكوين 3 عناقيد (مجموعات) من شخصين. نستخدم الآن العينة العشوائية البسيطة على هذه العناقيد. كما فعلنا سابقاً، لإيجاد احتمالية ظهور A في العينة:

$$P(A \text{ in sample}) = P(AB \text{ drawn}) = \frac{1}{3}$$

وبنفس النتيجة، فإن احتمالية ظهور أي شخص في العينة العشوائية هو $\frac{1}{3}$. نلاحظ أن الاختلاف الوحيد بين هذه الطريقة وطريقة العينة العشوائية البسيطة هو في العينات نفسها. مثلاً، في العينة العشوائية البسيطة احتمالية حصولنا على AB هو نفس احتمالية حصولنا على AC وهي $\frac{1}{15}$. ولكن في العينة العنقودية الاحتمالية هي:

$$P(AB) = \frac{1}{3}$$

$$P(AC) = 0$$

لأن A و C لا يظهران معاً في نفس المجموعة.

إذاً لماذا نستخدم العينة العنقودية؟ الطريقة فعالة جداً لأنها تسهل تجميع العينات. مثلاً، يعتبر من الأسهل إجراء استطلاع في مدينة مكونة من 100 شخص بدلاً من استطلاع آلاف الأشخاص موزعين على عدد مختلف من المدن والولايات. لهذا الكثير من مؤسسات التصويت هذه الأيام تستخدم العينات العنقودية لإجراء استطلاعاتها.

السلبية الوحيدة في العينة العنقودية هي أنها قد تظهر اختلاف Variance كبير في التوقع. يعني ذلك أننا نحتاج لأخذ عينات أكبر عند استخدام هذه الطريقة. ولاحظ أن في الواقع هذا أعقد بكثير من مثالنا السابق. 📝 📝

العينة الطبقية Stratified Sampling

كما يظهر من اسمها، في العينة الطبقية نقسم البيانات إلى طبقات، ثم نوجد عينة عشوائية بسيطة لكل طبقة. في كلا النوعين، الطبقية والعنقودية، نقسم البيانات إلى مجموعات، في العنقودية نستخدم العينة العشوائية البسيطة مرة واحدة، وفي العينة الطبقية نستخدمها أكثر من مرة (في كل مجموعة).

على نفس المثال السابق، يمكننا تقسيم الأشخاص الستة إلى الطبقتين التاليتين:

$$\text{Strata 1:} \quad \{A, B, C, D\}$$

$$\text{Strata 2:} \quad \{E, F\}$$

نستخدم العينة العشوائية البسيطة على كُل طبقة لنستخرج قيمة واحدة لنكون المجموعة المطلوبة والتي حجمها يساوي 2. الاحتمالات التي يمكن أن تحدث هي:

$$(A, E) \quad (A, F) \quad (B, E) \quad (B, F) \quad (C, E) \quad (C, F) \quad (D, E) \quad (D, F)$$

مرة أخرى، يمكننا إيجاد احتمالية وجود A في عينتنا العشوائية:

$$P(A \text{ in sample}) = P(A \text{ selected from Strata 1})$$

$$= \frac{2}{8}$$

$$= \frac{1}{4}$$

كما في العينة العنقودية، احتمالية $P(AB) = 0$ لأن A و B في نفس الطبقة.

كلا الطريقتين، العنقودية والطبقية، ينتجان نتائج مختلفة للاحتمالات بناءً على طريقة تقسيم البيانات فيهما. ويمكن ملاحظة أنه لا يجب أن تكون كل التقسيمات (الطبقات) متساوية الحجم. على سبيل المثال، يمكننا تقسيم الولايات المتحدة بناءً على الوظائف، ثم نأخذ عينات من كل طبقة بناءً على نسبة توزيع الوظائف، إذا كان فقط 0.01% منهم إحصائيين، فيجب أن نتأكد أن 0.01% من نسبة عينتنا تحتوي على إحصائيين.

إذا كنت تتوقع أن عينة الطبقات هي الأفضل للاستخدام للقيام بالعينة الحصصية Qouta Sampling فأنت على صواب. عينة الطبقات تسمح للباحث بالتأكد أن جميع الطبقات والمجموعات الفرعية ممثلة بشكل مناسب في العينة النهائية دون تدخل بشري لاختيار العينات. هذا قد يقلل أحياناً من الاختلاف في التوقع، ولكن العينة الطبقية قد تكون أصعب لأننا أحياناً لا نعرف حجم كل طبقة. 📝 📝

لماذا عينة الاحتمالات؟

تساعدنا عينة الاحتمالات في تحديد عدم يقيننا في تقدير أو تنبؤ. فقط بهذه الخطوة يمكننا اختبار فرضياتنا واستنتاجاتنا. كُن حذراً من أن يعطيك شخصاً قيمة عينة الاحتمالات أو مستويات الثقة في مشاريعهم دون شرح وافي وواضح عن طريقتهم في إيجاد العينات.

الآن بما أننا فهمنا عينة الاحتمالات، لنقارن بين العينة العشوائية البسيطة والبيانات الضخمة.

العينة العشوائية البسيطة × البيانات الضخمة

كما ذكرنا سابقاً، قد يبدو أسهل لنا استخدام بيانات أكثر للتخلص من الانحياز. صحيح أن جمع بيانات أكثر منطقياً يظهر نتائج غير متحيزة، ربما لن يكون الانحياز مشكلة بالنسبة لنا إذا ما قمنا فقط بجمع المزيد من البيانات.

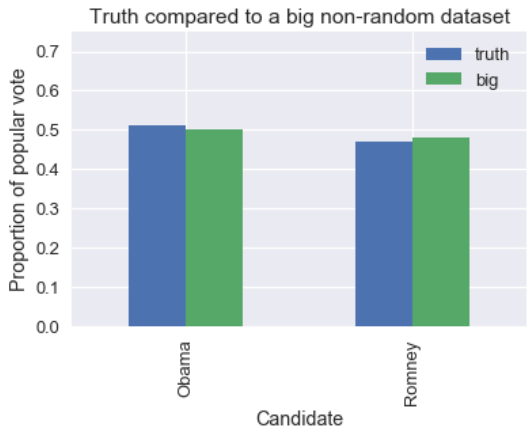
لنتخيل أننا منظمي استطلاعات ونريد توقع الرئيس القادم للولايات المتحدة الأمريكية في 2012، السباق كان بين "باراك أوباما" و "ميت رومني". بما أننا نعرف اليوم من فاز في هذا السباق، يمكننا أن نقارن بين توقعات العينة العشوائية البسيطة و توقعات البيانات الغير عشوائية الضخمة، يطلق عليها أحياناً البيانات الإدارية Administrative Datasets لأنها بيانات تُجمع من أنظمة إدارية.

سنقارن بين بيانات عشوائية بسيطه حجمها 400، وبيانات غير عشوائية ضخمة حجمها 60,000,000. بياناتنا الغير عشوائية حجمها أكثر ب150,000 مرة عن بياناتنا العشوائية!

في الرسم البياني التالي مقارنة بين النتائج الحقيقة لسباق الرئاسة والعينة الغير عشوائية. اللون الأزرق Truth تعني الأرقام الحقيقة التي حصل عليها كل مرشح، اللون الأخضر Big يظهر الأرقام من بياناتنا الضخمة:

```
total = 129085410
obama_true_count = 65915795
romney_true_count = 60933504
obama_true = obama_true_count / total
romney_true = romney_true_count / total

# 1 percent off
obama_big = obama_true - 0.01
romney_big = romney_true + 0.01

pd.DataFrame({
    'truth': [obama_true, romney_true],
    'big': [obama_big, romney_big],
}, index=['Obama', 'Romney'], columns=['truth', 'big']).plot.bar()
plt.title('Truth compared to a big non-random dataset')
```

```
plt.title('Truth compared to a big non-random dataset')
plt.xlabel('Candidate')
plt.ylabel('Proportion of popular vote')
plt.ylim(0, 0.75);
```

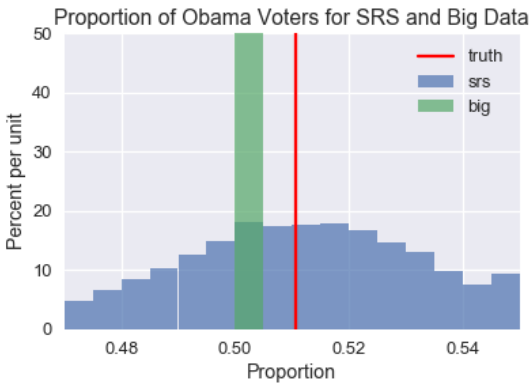


يمكن أن نرى أن البيانات الغير عشوائية الضخمة انحازت قليلاً للمرشح الجمهوري "ميت رومني"، كما كانت في تصويت غالوب عام 1948. لنرى تأثير هذا الانحياز، لنأخذ عينة عشوائية بسيطه من 400 شخص و عينه ضخمة غير عشوائية حجمها 60,000,000. سنحسب نسبة المصوتين للمرشح "باراك أوباما" في كل عينة:

```
srs_size = 400
big_size = 60000000
replications = 10000

def resample(size, prop, replications):
    return np.random.binomial(n=size, p=prop, size=replications) / size

srs_simulations = resample(srs_size, obama_true, replications)
big_simulations = resample(big_size, obama_big, replications)
```

ولرسم النتائج، بنفس الألوان السابقة، اللون الأزرق يعني البيانات العشوائية البسيطة، واللون الأخضر يعني نتائج توقع البيانات الضخمة، الخط الأحمر هي النتائج الحقيقية:

```
bins = bins=np.arange(0.47, 0.55, 0.005)
plt.hist(srs_simulations, bins=bins, alpha=0.7, normed=True, label='srs')
plt.hist(big_simulations, bins=bins, alpha=0.7, normed=True, label='big')

plt.title('Proportion of Obama Voters for SRS and Big Data')
plt.xlabel('Proportion')
plt.ylabel('Percent per unit')
plt.xlim(0.47, 0.55)
plt.ylim(0, 50)
plt.axvline(x=obama_true, color='r', label='truth')
plt.legend();
```



كما نرى، توزيع نتائج البيانات العشوائيه البسيطه مُتركز حول النتيجه الحقيقه للتصويت، بينما توزيع البيانات الغير عشوائيه الضخمه عكس ذلك، ولم ينتج أي تنبؤ صحيح لنسبة المصوتين الحقيقيه. إذا اردنا أن نكون مجالاً للثقه Confidence Intervals باستخدام البيانات الغير عشوائيه، فلن يحتوي أياً منها على نسبه التصويت الحقيقه. لجعل الأمر أكثر سوءًا، مجال الثقة سيكون نحيفاً جداً لأن العينه كبيره. وبالتأكيد ستظهر لنا توقعات خاطئة. 📝📝

في الحقيقة، عندما تكون طريقة إيجاد العينه لدينا مُنحازه فإن توقعنا عادة ما يكون في البدايه سيء، وكلما جمعنا المزيد من البيانات فأننا نؤكد خطأ نتائجنا. للقيام بتوقعات صحيحه حتى باستخدام عينات متحيزه قليلاً، العينه يجب أن يكون حجمها مقارب لحجم المجموعه، وهذا مطلب لا يمكن تنفيذه عادةً. **جودة البيانات تهم أكثر بكثير من حجمها**.

ملخص الفصل الثاني

قبل قبول نتائج تحليل البيانات، يجب أن نتحقق من جودة البيانات. يجب أن نسأل أنفسنا الأسئله التاليه:

- هل البيانات كامله (تحتوي على كامل المجموعه التي تهمنا)؟ إذا كانت كذلك، يمكننا إذاً إيجاد خصائص المجموعات دون الحاجة للإستدلال.
- إذا كانت البيانات عينه، كيف تم جمع العينه؟ للقيام بالإستدلال بشكل صحيح، يجب أن تكون البيانات قد جمعت باستخدام طريقه تستخدم الاحتمالات، ويتم شرحها بشكل كامل.
- ما هي التعديلات التي تمت على البيانات قبل إيجاد النتائج؟ هل أثرت هذه التعديلات على جودة البيانات؟

للحصول على معلومات أكثر ومقارنه بين العينات العشوائيه البسيطه والغير عشوائيه الضخمه، نقترح مشاهدة هذا الدرس بواسطة الإحصائي تشاو لي مينق.

البيانات المجدولة ومكتبة باندا‌ز

مقدمة

العمل مع البيانات المجدولة

البيانات المجدولة Tabular data تعتبر من أكثر أشكال البيانات استخداماً. في الفصلين السابقة استخدمنا مكتبة باندا‌ز Pandas للتعامل مع البيانات المجدولة، باندا‌ز تعتبر المكتبة الرائدة في هذا المجال. رغم أن الكتابة في باندا‌ز تبدو صعبة بعض الشيء، لكنها تحسن كثيراً من الأداء وتسهل التعامل مع البيانات المجدولة وتعتبر الأولى في المجال سواء في الأوساط العملية أو الأكاديمية.

من المهم أن تفهم الخطوات الأساسية للتعامل مع البيانات بدلاً من فهم تفاصيل كتابة الأوامر في باندا‌ز. مثلاً، فهم متى تستخدم خاصية الجمع Group by أكثر أهمية من معرفة كيفية كتابة الأمر في باندا‌ز. بما أن هذا الفصل يحتوي على الكثير من الأكواد البرمجية، ننصحك بالتركيز والقراءة أكثر من مرة: مرة لفهم الكود البرمجي، ومرة لفهم متى يجب عمل هذه الأمر.

لأننا سنغطي فقط أهم الأوامر المستخدم في باندا‌ز، يمكنك الرجوع دائماً لشرح المكتبة عندما تقوم بنفسك بتحليل البيانات.

سنبدأ أولاً بالحديث عن أشكال البيانات التي يمكن لباندا‌ز قراءتها. ثم سنتكلم عن أسماء الأعمدة، التقسيمات، التجميع، دالة التطبيق والنصوص في البيانات المجدولة.

هيكلة البيانات

هيكل وشكل البيانات يعتبر مهم لفهم وعرض البيانات. مثلاً، البيانات التي تكون على شكل **جداول** نعرضها في أعمدة وصفوف. على العكس **البيانات الهرمية**، كشجرة العائلة، تسمح لكل قيمة أن تحتوي على قيم أخرى بأشكال مختلفة. يوجد الكثير من الطرق لهيكلة وعرض البيانات، في هذا الكتاب سنعمل غالب الوقت على البيانات التي تظهر على شكل جداول.

صيغة ملف البيانات، توضح كيف تم حفظ البيانات على جهاز الكمبيوتر. مثلاً، ملف CSV (comma-separated values) يحتوي على بيانات يفصل بين كل قيمة والأخرى فاصلة (,). في هذا الكتاب سنطبق على عدة أنواع:

- CSV (Comma-Separated Values) و TSV (Tab-Separated Values). هذان النوعان عادةً ما تحتوي على بيانات بشكل جداول. في هذه الملفات كل سطر يمثل صف; ويفصل بين البيانات بعلامة الفاصلة (,) أو علامة Tab (\t). أول صف في هذه الملفات عادة ما يكون مخصص لأسماء الأعمدة.
- JSON (JavaScript Object Format) نوع آخر يستخدم بكثرة للتواصل بين تطبيقات وخوادم المواقع. ملفات JSON تكون ذات هيكل هرمي وتحتوي على مفاتيح Keys وقيم Values، عندما نريد قيمة معينة نبحث عنها بالمفتاح الخاص فيها، تشبه بشكل كبير القواميس Dictionary في لغة بايثون.
- XML (eXtensible Markup Language) و HTML (HyperText Markup Language) هي صيغ معروفة لحفظ البيانات على الإنترنت. كما في JSON، هذه الملفات هرمية وتتبع طريقة المفاتيح والقيم.

يوجد الكثير من الأدوات للتعامل مع أنواع مختلفة من صيغ الملفات، في هذا الكتاب سنقوم دائما بمعالجة البيانات وتحويلها إلى جداول. لماذا نجرب أنفسنا على هذا الخيار؟ أولاً، الكثير من البحوث قامت بعرض الطرق المناسبة لحفظ ومعالجة الجداول. ثانياً، الجداول تشبه كثيراً المصفوفات لذا استخدامها أسهل بكثير للقيام بالعمليات الرياضية. أخيراً، جداول البيانات هي الأكثر استخداماً.

في باندا‌ز يمكننا استخدام الدالة `pd.read_csv` لقراءة ملفات CSV و TSV، الملفات الأخرى لها دوال مختلفة، لذا نحتاج أولاً التحقق من نوع صيغة الملف قبل التعامل معه.

أدوات سطر الأوامر

جميع أجهزة الكمبيوتر توفر إمكانية الوصول لـ **مترجم شيل Shell Interpreter**، مثل `sh` أو `bash`. كما في مترجم نصوص بايثون، يسمح المترجم للمستخدم بكتابة الكود البرمجي وعرض نتيجته. مترجم شيل يقوم بعملياته مع الحاسب وملفاته، ولدية لغته الخاصة ودوال وطريقه مختلفة بالكتابة.

نستخدم المصطلح **أداة سطر الأوامر (command-line interface (CLI** للحديث عن الأوامر التي تستخدم في الشيل. رغم أننا سنتحدث عن القليل من أدوات سطر الأوامر في هذا الفصل، إلا أنه يوجد الكثير من الأدوات التي تسمح لنا بالعديد من العمليات التي يمكن فعلها على الحاسب.

ملاحظة: جميع أدوات سطر الأوامر مخصصة لشيل sh، المترجم الافتراضي على أجهزة ماكنتوش ولينكس. تم استخدام Jupyter والذي يستخدم نفس الأداة. أجهزة ويندوز تستخدم مترجم سطر أوامر آخر وبعض الأوامر المستخدمة في هذا الكتاب قد لا تعمل على ويندوز.

يمكن استخدام أوامر sh على أجهزة ويندوز عن طريق Gitbash.

بما أن أداة جوبتر تستخدم نفس مترجم شيل، فبإمكاننا استخدام الأوامر في خلايا الأداة. إذا أضيفت علامة تعجب (!) قبل الحرف، فإن الأداة تفهم أن هذا الأمر هو أمر لمترجم شيل فيتم إرسال الأمر له. مثلاً الأمر `ls` يعرض لنا جميع الملفات في المجلد الذي نعمل فيه حالياً:

```
!ls
```

```
babynames.csv                       pandas_indexes.ipynb
others                              pandas_intro.ipynb
pandas_apply_strings_plotting.ipynb pandas_structure.ipynb
pandas_grouping_pivoting.ipynb
```

أدوات ودوال سطر الأوامر مثل `ls` عادة ما تأخذ **متغيرات**، تماماً كما تأخذ دوال بايثون متغيرات. في sh، نقوم بوضع المتغيرات بين مسافات، وليس أقواس كما في بايثون. كتابة الأمر `ls` ثم اسم المجلد يظهر لنا الملفات داخل ذلك المجلد:

```
!ls others
```

```
babies.data
```

عندما نجد الملف الذي نبحث عنه، نستخدم أدوات سطر الأوامر للتحقق من هيكلة. الأمر `head` يظهر لنا بعض الأسطر في الملف، يمكننا من إلقاء نظرة سريعة على محتوى الملف:

لتحميل البيانات، اضغط هنا.

```
!head others/babies.data
```

```
bwt gestation parity age height weight smoke
120 284   0  27  62 100   0
113 282   0  33  64 135   0
128 279   0  28  64 115   1
123 999   0  36  69 190   0
108 282   0  23  67 125   1
136 286   0  25  62  93   0
138 244   0  33  62 178   0
132 245   0  23  65 140   0
120 289   0  25  62 125   0
```

العدد الافتراضي للأسطر الذي يعرضها الأمر `head` هي أول 10 أسطر في الملف، يمكننا عرض آخر 10 أسطر باستخدام `tail`:

```
!tail others/babies.data
```

```
103 278   0  30  60  87   1
118 276   0  34  64 116   0
127 290   0  27  65 121   0
132 270   0  27  65 126   0
113 275   1  27  60 100   0
128 265   0  24  67 120   0
130 291   0  30  65 150   1
125 281   1  21  65 110   0
117 297   0  38  65 129   0
```

يمكننا طباعة كل محتوى الملف باستخدام الأمر `cat`. انتبه عند استخدام هذا الأمر، لأن طباعة الملفات ذات المحتوى الكثير قد يؤثر على الشيل:

```
!cat others/text.txt
```

```
"city","zip","street"
"Alameda","94501","1220 Broadway"
"Alameda","94501","429 Fair Haven Road"
"Alameda","94501","2804 Fernside Boulevard"
"Alameda","94501","1316 Grove Street"
```

في أحيان عديدة، استخدام `head` و `tail` يكفي لمعرفة هيكلة الملف. مثلاً يمكننا رؤية أن ملف `babynames.csv` يستخدم صيغة CSV:

لتحميل البيانات، اضغط هنا.

```
!head babynames.csv
```

```
Name,Sex,Count,Year
Mary,F,9217,1884
Anna,F,3860,1884
Emma,F,2587,1884
Elizabeth,F,2549,1884
Minnie,F,2243,1884
Margaret,F,2142,1884
Ida,F,1882,1884
Clara,F,1852,1884
Bertha,F,1789,1884
```

يمكننا قراءة ملفات CSV باستخدام باندا والدالة `pd.read_csv`:

```
# تستخدم pd بشكل كبير كأختصار ل Pandas. سنستخدم دائماً pd كأسم مختصر للمكتبة باندا.
import pandas as pd

pd.read_csv('babynames.csv')
```

	Name	Sex	Count	Year
0	Mary	F	9217	1884
1	Anna	F	3860	1884
2	Emma	F	2587	1884
...	...	...	...	...
1891891	Verna	M	5	1883
1891892	Winnie	M	5	1883
1891893	Winthrop	M	5	1883

1891894 rows × 4 columns

حجم الملفات

لاحظ أن قراءة الملف `babynames.csv` نتج عنها DataFrame تحتوي على أكثر من 2 مليون سطر. جميع أجهزة الكمبيوتر لديها قدرات محدودة لا تستطيع تجاوزها. وقد يحدث أن تصل لهذه المراحل مرات عديدة ويبدأ جهازك بالعمل ببطيء بسبب أن كثير من البرامج تعمل في نفس الوقت. نحاول دائماً على عدم الوصول لهذه المرحلة أثناء عملنا مع البيانات.

DataFrame: جدول ثنائي الأبعاد، يستخدم كثيراً مع البيانات المجدولة ويحتوي على أعمدة وصفوف. لمعلومات أكثر هنا.

في كثير من الحالات، نستخدم بيانات للتحليل يتم تجميعها من الإنترنت. وتشغل هذه الملفات جزء من **مساحة التخزين** في الجهاز. يحتاج بايثون للتعامل وتعديل هذه البيانات لقراءة البيانات عن طريق **ذاكرة التخزين العشوائية RAM**. حجم الذاكرة العشوائية يكون غالباً أصغر بكثير من حجم مساحة التخزين.

كلاهما، مساحة التخزين وذاكرة التخزين العشوائية، يستخدمون **البايت Byte** لقياس مساحتها. تقريباً، كل حرف في ملف نصي عبارة عن بايت واحد. مثلاً ملف يحتوي على 177 حرفاً يشغل مساحة 177 بايت من مساحة التخزين.

بالطبع الكثير من البيانات التي نعمل عليها اليوم تحتوي على الكثير من الحروف، لوصف حجم الملفات ذات الحجم الكبير، نستخدم التالي:

المضاعف	الرمز	عدد البايت
Kibibyte	KiB	$1024 = 2^{10}$
Mebibyte	MiB	$1024^2 = 2^{20}$
Gibibyte	GiB	$1024^3 = 2^{30}$
Tebibyte	TiB	$1024^4 = 2^{40}$
Pebibyte	PiB	$1024^5 = 2^{50}$

مثلاً، ملف يحتوي على 52428800 حرفاً، يشغل مساحة 52428800 بايت = 50 ميبيبايت = 50 MiB.

$$\frac{52428800 \text{ byte}}{1024} = 51200 \text{ KiB}$$

$$\frac{52428800 \text{ byte}}{1024^2} = \frac{52428800}{1048576} = 50 \text{ MiB}$$

لماذا نستخدم مضاعفات 1024 بدلاً من 1000؟ لأن جميع أجهزة الكمبيوتر تستخدم نظام العد الثنائي Binary Numbers وفيها مضاعفات 2 أسهل للاستخدام. سترى أيضاً الوحدات المعروفة دولياً تستخدم لحساب أحجام الملفات، مثل كيلو بايت، ميقا بايت وقيقا بايت. للأسف تستخدم هذه الوحدات بشكل غير متفق عليه. أحياناً يقصد بالكيلو بايت 1000 بايت، ومرات أخرى يقصد 1024 بايت. للتخلص من هذا الالتباس، سنستخدم الكيبي، ميبي، والجيبي بايت والتي تصف بشكل دقيق مضاعفات 1024.

متى يكون آمناً قراءة الملف؟

قبل البدء في تحليل البيانات، نحتاج أن نتأكد أن أحجام الملفات التي سنعمل عليها يمكن التحكم بها. ونستخدم لذلك أداة سطر الأوامر والأمرين `ls` و `du`:

```
!ls others
```

```
babies.data text.txt
```

أوامر سطر الأوامر يمكن إضافة بعض المتغيرات عليها لتقدم لنا مزيداً من المعلومات. مثلاً، إضافة `-l` للأمر يعطينا التالي:

```
!ls -l others
```

```
total 80
-rw-r--r--@ 1 sam  staff  34654 Dec 19 13:34 babies.data
-rw-r--r--  1 sam  staff    177 Dec 19 13:37 text.txt
```

العمود الخامس من النتيجة السابقة للأمر تعطينا حجم الملف بالبايت. ونرى أن الملف `babies.data` يشغل 34654 بايت. لجعل القيمة أكثر سهولة للقراءة نستخدم الأمر `-h`:

```
!ls -l -h others
```

```
total 80
-rw-r--r--@ 1 sam  staff    34K Dec 19 13:34 babies.data
-rw-r--r--  1 sam  staff   177B Dec 19 13:37 text.txt
```

نرى الآن أن الملف يشغل 34 كيبي بايت. رغم أن الملف `babynames.csv` يحتوي على أكثر من 2 مليون سطر، إلا أنة يشغل فقط 30 ميبي بايت.

```
!ls -l -h
```

```
total 62896
-rw-r--r--  1 sam  staff    30M Aug 10 22:35 babynames.csv
drwxr-xr-x  4 sam  staff   128B Dec 19 13:37 others
-rw-r--r--  1 sam  staff   118K Sep 25 17:13 pandas_apply_strings_plotting.ipynb
-rw-r--r--  1 sam  staff    34K Sep 25 17:13 pandas_grouping_pivoting.ipynb
```

```
-rw-r--r--  1 sam  staff    32K Dec 19 13:07 pandas_indexes.ipynb
-rw-r--r--  1 sam  staff   2.1K Dec 19 13:23 pandas_intro.ipynb
-rw-r--r--  1 sam  staff    23K Dec 19 13:44 pandas_structure.ipynb
```

حجم المجلدات

في بعض الأحيان نريد معرفة حجم مجلد بدلاً من حجم الملفات. نلاحظ أن `ls` لا تعطي حجم المجلدات بشكل صحيح. مثلاً هنا حجم المجلد `others` هو 128 بايت.

```
!ls -l -h
```

```
total 62896
-rw-r--r--  1 sam  staff    30M Aug 10 22:35 babynames.csv
drwxr-xr-x  4 sam  staff   128B Dec 19 13:37 others
-rw-r--r--  1 sam  staff   118K Sep 25 17:13 pandas_apply_strings_plotting.ipynb
-rw-r--r--  1 sam  staff    34K Sep 25 17:13 pandas_grouping_pivoting.ipynb
-rw-r--r--  1 sam  staff    32K Dec 19 13:07 pandas_indexes.ipynb
-rw-r--r--  1 sam  staff   2.1K Dec 19 13:23 pandas_intro.ipynb
-rw-r--r--  1 sam  staff    23K Dec 19 13:44 pandas_structure.ipynb
```

ولكن المجلد يحتوي على ملفات أكبر من 128 بايت:

```
!ls -l -h others
```

```
total 80
-rw-r--r--@ 1 sam  staff    34K Dec 19 13:34 babies.data
-rw-r--r--  1 sam  staff   177B Dec 19 13:37 text.txt
```

لحساب حجم الملفات داخل المجلدات بشكل صحيح، نستخدم الأمر `du`. بشكل تلقائي، الأداة `du` تظهر حجم الملفات بصيغة المجموعات أو الكُتل blocks: 📝

```
!du others
```

```
80  others
```

لعرض الحجم بالبايت نستخدم `-h`:

```
!du -h others
```

```
40K  others
```

في العادة نضيف الأمر `-s` لـ `du` لعرض جميع أحجام الملفات والمجلدات معاً. النجمة * في `others/*` تعني أن على `du` إظهار معلومات جميع الملفات داخل المجلد `other`:

```
!du -sh others/*
```

```
36K         others/babies.data
4.0K    others/text.txt
```

الذاكرة غير المباشرة

قاعدة عامة، قراءة ملف باستخدام مكتبة بانداز عادة ما يحتاج أن تكون ذاكرة الجهاز ضعف حجم الملف نفسه. مثلاً قراءة ملف بحجم 1 جيبي بايت يحتاج أن يكون الجهاز يحتوي على ذاكرة بحجم 2 جيبي بايت.

جميع البرامج تستخدم الذاكرة للعمل، بما في ذلك نظام التشغيل، متصفح الإنترنت، وكذلك جوبتر نوت بوك. جهاز ب 4 جيبي بايت قد لا يحتوي سوى على 1 جيبي بايت متوفرة للاستخدام إذا كان الكثير من البرامج يعملون في نفس الوقت. بحجم 1 جيبي بايت من الذاكرة المتوفرة للاستخدام من الصعب على بانداز قراءة ملف بحجم 1 جيبي بايت.

ملخص

في هذا الجزء، تعرفنا على هيكلة البيانات وجدولتها والتي سنستخدمها كثيراً في بقية الكتاب. كذلك عرضنا بعض الأوامر في أداة سطر الأوامر مثل ، `ls` `head` ،`du` و `tail`. هذه الأوامر تساعدنا على فهم شكل الملفات التي سنعمل عليها. استخدمنا نفس الأدوات للتحقق من قدرة بانداز على قراءة الملفات. عندما تقرأ بانداز الملف، سيكون لدينا DataFrame جاهز للتحليل.

الفهرسة، التقسيم والترتيب

مقدمة

في ما تبقى من هذا الفصل سنعمل على نفس بيانات أسماء الأطفال التي سبق أن تحدثنا عنها في الفصل الأول. سنطرح سؤالاً، ثم نقسم السؤال إلى عدة أقسام، ونترجم كل جزء إلى كود برمجي في بايثون باستخدام مكتبة بانداز.

أولاً نقوم باستدعاء مكتبة بانداز:

```
import pandas as pd
```

بعد أن استدعيناها، يمكننا الآن قراء البيانات باستخدام الدالة `pd.read_csv` (معلومات أكثر عن الدالة هنا):

```
baby = pd.read_csv('babynames.csv')
baby
```

	Name	Sex	Count	Year
0	Mary	F	9217	1884
1	Anna	F	3860	1884
2	Emma	F	2587	1884
...	...	...	...	...
1891891	Verna	M	5	1883
1891892	Winnie	M	5	1883
1891893	Winthrop	M	5	1883

1891894 rows × 4 columns

لاحظ، لكي يعمل الكود البرمجي في الأعلى يجب أن يكون ملف `babynames.csv` متواجد في نفس المجلد الذي نعمل عليه، يمكننا التحقق من الملفات الموجودة في المجلد باستخدام الأمر `ls`:

```
!ls
```

```
babynames.csv                       pandas_indexes.ipynb
others                              pandas_intro.ipynb
pandas_apply_strings_plotting.ipynb pandas_structure.ipynb
pandas_grouping_pivoting.ipynb
```

عندما نستخدم مكتبة بانداز لقراءة البيانات، يتم تحويلها بشكل أوتوماتيكي إلى DataFrame. لكل عمود اسم، مثلاً (‘Name’، ‘Sex’، ‘Count’، ‘Year’) ولكل سطر رقم مفهرس، مثلاً (0، 1، 2، ...، 1891893).

مسميات الأعمدة في ال DataFrame يطلق عليها Labels ولكل عمود رقم فهرسي Index، اسم العمود والرقم الفهرسي تسهل علينا الكثير في التعامل مع البيانات. الصفوف أيضاً لها أرقام فهرسيه Indexes.

لنستخدم بانداز للإجابة على السؤال التالي:

ما هي أسماء الأطفال الأكثر شهره في عام 2016؟

نبدأ أولاً بتقسيم المشكلة

لتسهيل الإجابة، يمكننا تقسيم السؤال إلى جزأين:

- تقسيم البيانات وسحب بيانات عام 2016 فقط.
- الترتيب تنازلياً حسب التكرار بإستخادم عمود Count.

التقسيم باستخدام .loc

لاختيار جزء من DataFrame، نستخدم .loc المتغير الأول فيها هو اسم السطر والثاني هو اسم العمود:

```
# السطر رقم 1
# العمود Name
baby.loc[1, 'Name']
```

```
'Anna'
```

لسحب أكثر من سطر أو عمود، نستخدم :

```
# نطلب هنا السطر 1 إلى 5
# والأعمدة من Name حتى Count
baby.loc[1:5, 'Name':'Count']
```

	Name	Sex	Count
1	Anna	F	3860
2	Emma	F	2587
3	Elizabeth	F	2549
4	Minnie	M	2243
5	Margaret	M	2142

أحياناً، نريد عمود واحد فقط من ال DataFrame، نستخدم التالي:

```
baby.loc[:, 'Year']
```

```
0          1884
1          1884
2          1884
           ...
1891891    1883
1891892    1883
1891893    1883
Name: Year, Length: 1891894, dtype: int64
```

عندما نختار عمود واحد فقط، تتحول ال DataFrame إلى مجموعة Series. المجموعة هي مصفوفة NumPy أحادية البعد يمكن أن نقوم بعمليات حسابية على جميع محتوياتها:

```
baby.loc[:, 'Year'] * 2
```

```
0          3768
1          3768
2          3768
           ...
1891891    3766
1891892    3766
1891893    3766
Name: Year, Length: 1891894, dtype: int64
```

لإختيار أعمدة معينة فقط، يمكننا تمرير مصفوفة بداخل .loc:

```
baby.loc[:, ['Name', 'Year']]
```

	Name	Year

	Name	Year
0	Mary	1884
1	Anna	1884
2	Emma	1884
...	...	...
1891891	Verna	1883
1891892	Winnie	1883
1891893	Winthrop	1883

1891894 rows × 2 columns

طريقة أخرى لإيجاد بيانات عمود واحد:

```
# اختصاراً ل baby.loc[:, 'Name']
baby['Name']
```

```
0             Mary
1             Anna
2             Emma
             ...
1891891       Verna
1891892      Winnie
1891893    Winthrop
Name: Name, Length: 1891894, dtype: object
```

```
# اختصاراً ل baby.loc[:, ['Name', 'Count']]
baby[['Name', 'Count']]
```

	Name	Count
0	Mary	9217
1	Anna	3860
2	Emma	2587
...	...	...
1891891	Verna	5
1891892	Winnie	5
1891893	Winthrop	5

1891894 rows × 2 columns

التقسيم المشروط

لإيجاد الصفوف لسنة 2016 فقط، يجب أن نجد أولاً مجموعة Series تحتوي على True و False (صح أو خطأ)، فيها True لكل قيمة نريدها و False للقيم التي لا نريدها. مجموعة السنوات:

```
# لإيجاد السنوات
baby['Year']
```

```
0          1884
1          1884
2          1884
           ...
1891891    1883
1891892    1883
1891893    1883
Name: Year, Length: 1891894, dtype: int64
```

```
# مقارنة كل سطر مع السنة 2016
baby['Year'] == 2016
```

```
0          False
1          False
2          False
           ...
1891891    False
1891892    False
1891893    False
Name: Year, Length: 1891894, dtype: bool
```

بهذه الطريقة تتم المقارنه بين كل سطر والسنه 2016، في حال كان السطر يساوي 2016 يظهر True والعكس إذا كانت لا تساوي False، يمكننا دمجها مع `.loc`:

```
# نريد التقسيم بالإسطر، لذا نضع شرطنا في المتغير الأول في .loc
baby_2016 = baby.loc[baby['Year'] == 2016, :]
baby_2016
```

	Name	Sex	Count	Year
1850880	Emma	F	19414	2016
1850881	Olivia	F	19246	2016
1850882	Ava	F	16237	2016
...	...	...	...	...
1883745	Zyahir	M	5	2016
1883746	Zyel	M	5	2016
1883747	Zylyn	M	5	2016

32868 rows × 4 columns

ترتيب الصفوف

بعد أن وجدنا فقط أسماء أطفال 2016، الخطوة التالية هي ترتيب الأسماء تنازلياً بناءاً على عمود Count. سنستخدم الدالة `sort_values()`:

```
sorted_2016 = baby_2016.sort_values('Count', ascending=False)
sorted_2016
```

	Name	Sex	Count	Year
1850880	Emma	F	19414	2016
1850881	Olivia	F	19246	2016
1869637	Noah	M	19015	2016
...	...	...	...	...
1868752	Mikaelyn	F	5	2016
1868751	Miette	F	5	2016
1883747	Zylyn	M	5	2016

32868 rows × 4 columns

القيمة التلقائية للمتغير Ascending والذي يعني الترتيب تصاعدياً هي True، وإذا رغبنا ترتيب القيم تنازلياً، نضيف False للمتغير. معلومات أكثر هنا.

الآن، سنستخدم `.iloc` لإيجاد أول خمس أسطر من ال DataFrame. تعمل `.iloc` تماماً مثل `.loc` ولكن تأخذ قيم رقمية بدلاً من نصية / أسماء أعمدة. ولا تحسب آخر رقم كما في بايثون.

```
# إيجاد السطر رقم 0 والعمود رقم 0
sorted_2016.iloc[0, 0]
```

```
'Emma'
```

```
# إيجاد أول خمس أسطر
sorted_2016.iloc[0:5]
```

	Name	Sex	Count	Year
1850880	Emma	F	19414	2016
1850881	Olivia	F	19246	2016
1869637	Noah	M	19015	2016
1869638	Liam	M	18138	2016
1850882	Ava	F	16237	2016

الملخص

لدينا الآن أكثر خمس أسماء شهره في عام 2016، وتعلمنا كيف نستخدم الخمس دوال التالية في مكتبة بانداز:

الدالة	العملية
`pd.read_csv()`	قراءة ملف CSV
`loc.` أو `iloc.`	التقسيم باستخدام أسماء الأعمدة أو ترتيبها الرقمي
استخدام True/False داخل الدالة `loc.`	التقسيم الشرطي
`sort_values()`	الترتيب

التجميع والجداول المحورية

في هذا الجزء، سنجيب على السؤال التالي:

ما هي الأسماء الأكثر شهره بين الذكور والإناث كل سنه؟

تقسيم المشكلة

نلاحظ أن السؤال مشابه للسؤال في الجزء السابق؛ السؤال السابق يريد فقط أسماء سنوات 2016 بينما هذا السؤال يشمل جميع السنوات. نقسم السؤال إلى قسمين:

- نجمع الأسطر حسب عامودي السنة Year و الجنس Sex.
- لكل مجموعة، أبحث عن الاسم الأكثر شهره.

معرفة الدوال التي تستخدم في حل مشكلة قد يبدو صعباً. عادةً، عن العمل بخطوات طويلة معقدة تشعر بأنة قد يكون هناك طريقة أسهل لفعل الخطوات هذه، إذا لم نعلم أنه يجب أن نقوم بالجمع مباشره، فيمكن أن نكتب خطوات كالتالي:

- الدوران Loop بين كل سنة على حدة.
- لكل سنة، الدوران في عمود الجنس.
- لكل سنة وجنس، أوجد الاسم الأكثر شهره.

في بانداز، هناك دائماً طريقة أفضل من الدوران. الدوران على القيم في ال DataFrame يجب أن يستبدل بالتجميع.

التجميع

للتجميع في بانداز، نستخدم الدالة `groupby()`:

```
baby.groupby('Year')
```

```
<pandas.core.groupby.DataFrameGroupBy object at 0x1a14e21f60>
```

عند استخدام الدالة `groupby()` تعود لنا بكائن Object من نوع `DataFrameGroupBy`. يمكننا استخدام `agg()` مع دالة تجميع على هذا الكائن للحصول على نتيجة يمكن قرائتها:

الدالة `agg()` تسمح لنا باستخدام دوال على جميع الأعمدة.

```
# دالة تجميع هي أي دالة تقبل متغير، هنا مثلاً أستخدمنا مجموعة/مصفوفة، وينتج عنها نتيجة واحدة (رقم مثلاً)، هنا تعود طول المجموعة
def length(series):
    return len(series)
```

```
# حساب عدد الأصفف في كل سنة
baby.groupby('Year').agg(length)
```

	Name	Sex	Count
Year			
1880	2000	2000	2000
1881	1935	1935	1935
1882	2127	2127	2127
...	...	...	...
2014	33206	33206	33206
2015	33063	33063	33063
2016	32868	32868	32868

137 rows × 3 columns

تلاحظ أيضاً، ان الدالة التي عرفناها بأسم `length` تستدعي دالة أخرى بأسم `len`، لتبسيط الكود البرمجي يمكننا عمل التالي:

```
baby.groupby('Year').agg(len)
```

	Name	Sex	Count
Year			
1880	2000	2000	2000
1881	1935	1935	1935
1882	2127	2127	2127
...	...	...	...
2014	33206	33206	33206
2015	33063	33063	33063
2016	32868	32868	32868

137 rows × 3 columns

تم تطبيق التجميع على كل الأعمدة في ال DataFrame، نتج عن ذلك تكرار في حساب النتائج. يمكننا تحديد نتائج معينة عبر تقسيم الأعمدة قبل التجميع.

```
year_rows = baby[['Year', 'Count']].groupby('Year').agg(len)
year_rows

# طريقة أخرى لتقصير هذا الكود البرمجي عن طريق عمل التالي:
#
# year_counts = baby[['Year', 'Count']].groupby('Year').count()
#
# بانداز قامت بإختصار الكثير من دوال التجميع مثل count, sum و mean.
```

	Count
Year	
1880	2000
1881	1935
1882	2127
...	...
2014	33206
2015	33063
2016	32868

137 rows × 1 columns

لاحظ أن أرقام الأسطر Indexes تم تغيرها للسنوات، يمكننا الآن التقسيم حسب السنة بواسطة `.loc` كما تعلمنا سابقاً:

```
# إيجاد كل عشرين سنه بدءا من 1880
year_rows.loc[1880:2016:20, :]
```

	Count

	Count
Year	
1880	2000
1900	3730
1920	10755
1940	8961
1960	11924
1980	19440
2000	29764

التجميع بأكثر من عمود

كما تعلمنا في المادة السابقة داتا 8، يمكننا التجميع بأكثر من عمود لإيجاد نتائج بناءاً عليهم. لفعل ذلك، نقوم بتمرير قائمة بأسماء الأعمدة كمتغيرات للدالة `.groupby()`:

```
grouped_counts = baby.groupby(['Year', 'Sex']).sum()
grouped_counts
```

		Count
Year	**Sex**	
1880	F	90992
	M	110491
1881	F	91953
...	...	...
2015	M	1907211
2016	F	1756647
	M	1880674

274 rows × 1 columns

الكود البرمجي السابق يحسب مجموع الأطفال المولودين كل سنة لكل جنس. لنقوم الآن بجمع أكثر من عمود لحساب الأسماء الأكثر شهره لكل جنس وسنة. بما أن البيانات مرتبة تنازلياً بناءًا على العمود Count، يمكننا تعريف دالة تجميع تجلب لنا أول قيمة في كل مجموعة. (إذا كانت البيانات غير مرتبه، نستخدم الدالة `sort_values()` أولاً):

```
# (الأسم الأكثر شهره هو أول قيمة في المجموعة (بما أنها مرتبة من الأعلى للأدنى
def most_popular(series):
    return series.iloc[0]

baby_pop = baby.groupby(['Year', 'Sex']).agg(most_popular)
baby_pop
```

		Name	Count
Year	**Sex**		
1880	F	Mary	7065
	M	John	9655
1881	F	Mary	6919
...	...	...	...
2015	M	Noah	19594
2016	F	Emma	19414
	M	Noah	19015

274 rows × 2 columns

لتوضيح الكود البرمجي السابق، يمكن مشاهدة نتيجة الأمر `baby.groupby(['Year', 'Sex'])` باضافة `.apply(print)` ليصبح الكود بشكل كامل `baby.groupby(['Year', 'Sex']).apply(print)` النتيجة هنا مجموعة Series بأحجام مختلفه لكل سنة على حدة. تحتوي فيها على 4 أعمدة `Year`، `Count`، `Sex`، `Name` مثال لسنة 1880:

	Name	Sex	Count	Year
0	Mary	F	7065	1880
1	Anna	F	2604	1880

	Name	Sex	Count	Year
2	Emma	F	2003	1880
...	...	...	...	...
1884687	Verona	M	5	1880
1884688	Vertie	M	5	1880
1884689	Wilma	M	5	1880

عند استخدام الدالة التجميعية التي قمنا بتعريفها `most_popular` تقوم الدالة بإيجاد أول قيمة / سطر في المجموعة والذي يحتوي على عامودين `Name` و `Count`، يتم تجميعها مع العامودين `Year` و `Sex` لتتكون لنا نتيجة المطلوبة.

لاحظ أن التجميع بأكثر من عمود ينتج أكثر من مسمى للسطر. يسمى ذلك بالأسطر متعددة المستويات. يجب أن تعرف أن `.loc` تقبل مصفوفة من نوع صفوف Tuples بأسماء الأسطر بدلاً من قيمة واحدة:

```
baby_pop.loc[(2000, 'F'), 'Name']
```

```
'Emily'
```

ولكن لاستخدام `.iloc`، يمكننا إرسال الأرقام دون مصفوفة صفوف Tuples، كما فعلنا بالمثال السابق، لأن الدالة تقبل الأرقام فقط:

```
# السطر من 10 حتى 15، وجميع الأعمدة
baby_pop.iloc[10:15, :]
```

		Name	Count
Year	**Sex**		
1885	F	Mary	9128
	M	John	8756
1886	F	Mary	9889
	M	John	9026
1887	F	Mary	9888

الجداول المحورية

إذا جمعنا البيانات بناءاً على عامودين، فمن الأفضل استخدام الجداول المحورية Pivot Tables لتمثيل البيانات كونها تظهر لنا النتائج بشكل أكثر وضوحاً. لبناء الجداول المحورية نستخدم الدالة `pd.pivot_table()`: 📝

```
pd.pivot_table(baby,
               index='Year',          # عمود الأسم/الفهرس للأسطر
               columns='Sex',         # الأعمدة
               values='Name',         # القيم داخل الأعمدة
               aggfunc=most_popular) # دالة التجميع
```

Sex	F	M
Year		
1880	Mary	John
1881	Mary	John
1882	Mary	John
...	...	...
2014	Emma	Noah
2015	Emma	Noah
2016	Emma	Noah

137 rows × 2 columns

لنقارنها بنتيجة `baby_pop` عندما استخدامنا الدالة `.groupby()`، يمكننا ملاحظة أن الأعمدة أصبحت مقسمة حسب الجنس بدلاً من أن كانت مفهرسة Index:

```
baby_pop
```

		Name	Count
Year	**Sex**		
1880	F	Mary	7065
	M	John	9655
1881	F	Mary	6919
...	...	...	...
2015	M	Noah	19594
2016	F	Emma	19414
	M	Noah	19015

274 rows × 2 columns

الملخص

لدينا الآن أكثر الأسماء شهره لكل جنس في كُل سنة، وتعلمنا كيف نستخدم الدوال التالية في مكتبة بانداز:

الدالة	العملية
`(f.groupby(label`	التجميع
`([df.groupby([label1, label2`	تجميع أكثر من عمود
`(df.groupby(label).agg(func`	التجميع واستخدام الدوال على الأعمدة
`()pd.pivot_table`	الجداول المحورية

دالة التطبيق، النصوص والرسم البياني

سنجيب في هذا الجزء على السؤال التالي:

هل يمكننا استخدام آخر حرف من الاسم لتوقع جنس الطفل؟

سنستخدم نفس البيانات `babynames.csv`:

```
baby = pd.read_csv('babynames.csv')
baby.head()
# الدالة .head() تظهر لنا أول خمس أسطر في البيانات
```

	Name	Sex	Count	Year
0	Mary	F	9217	1884
1	Anna	F	3860	1884
2	Emma	F	2587	1884
3	Elizabeth	F	2549	1884
4	Minnie	F	2243	1884

تحديد أجزاء المشكلة

توجد طُرق كثيره للتنبؤ، لكن سنستخدم هنا الرسم البياني. سنقسم المشكلة إلى الخطوات التالية:

- إيجاد آخر حرف من كل اسم.
- تجميع آخر حرف مع الجنس، التجميع وإيجاد مجموع الأطفال.
- رسم النتائج لكل جنس وآخر حرف.

دالة التطبيق

تحتوي المجموعات Series في بانداز على دالة `.apply()`، التي تستقبل دوال وتطبق هذه الدالة على جميع القيم:

```
names = baby['Name']
names.apply(len)
```

```
0          4
1          4
2          4
          ..
1891891    5
1891892    6
1891893    8
Name: Name, Length: 1891894, dtype: int64
```

في المثال السابق، استخدم الكاتب `.apply()` لإيجاد طول كل اسم في المجموعة عن طريق أولاً وضع الأسماء في متغير `names` ثم تطبيق دالة `len` على جميع القيم.

لاستخراج آخر حرف من كل اسم، سنقوم بكتابة دالة وتمريرها في دالة التطبيق `.apply()`:

```
def last_letter(string):
    return string[-1]

names.apply(last_letter)
```

```
0          y
1          a
2          a
          ..
1891891    a
1891892    e
1891893    p
Name: Name, Length: 1891894, dtype: object
```

أستخدم الكاتب في دالة `last_letter` القيمة `string[-1]` والتي تعني آخر حرف في النص. لمزيد من المعلومات عن التعامل مع النصوص هنا و هنا.

التلاعب بالنصوص

رغم مرونة الدالة `.apply()`، إلا أنه من الأسرع استخدام الدوال الموجودة في باندا للتعامل مع النصوص. يوجد الكثير من الدوال الموجودة للتعامل مع النصوص باستخدام `.str` في المجموعات:

```
names = baby['Name']
names.str.len()
```

```
0          4
1          4
2          4
          ..
1891891    5
1891892    6
1891893    8
Name: Name, Length: 1891894, dtype: int64
```

ويمكننا استخراج آخر حرف من كل اسم بالطريقة التالية:

```
names.str[-1]
```

```
0          y
1          a
2          a
          ..
1891891    a
1891892    e
1891893    p
Name: Name, Length: 1891894, dtype: object
```

نقترح قراءة المزيد عن دوال النصوص في باندز عن طريق الرابط هنا.

يمكننا الآن إضافة عمود آخر حرف لبيانات الأطفال لدينا:

```
baby['Last'] = names.str[-1]
baby
```

	Name	Sex	Count	Year	Last
0	Mary	F	9217	1884	y
1	Anna	F	3860	1884	a
2	Emma	F	2587	1884	a
...	...	...	...	...	...
1891891	Verna	M	5	1883	a
1891892	Winnie	M	5	1883	e
1891893	Winthrop	M	5	1883	p

1891894 rows × 5 columns

التجميع

لإيجاد توزيع الجنس بناءاً على آخر حرف، نحتاج لجمع العامودين Last و Sex:

```
# استخدمنا طريقة مختصرة بدلاً من: baby.groupby(['Last', 'Sex']).agg(np.sum)
baby.groupby(['Last', 'Sex']).sum()
```

		Count	Year
Last	**Sex**		
a	F	58079486	915565667
	M	1931630	53566324
b	F	17376	1092953
...	...	...	...
y	M	18569388	114394474
z	F	142023	4268028
	M	9649274	19015

52 rows × 2 columns

لاحظ أنه تم جمع عمود السنة، لأنه يتم جمع أي عمود لا يتم تمريرة لدالة التجميع .groupby(). ولحل هذه المشكلة يمكننا تحديد الأعمدة التي نرغب باستخدامها قبل تمريرها لدالة التجميع:

```
# عندما تكون الأسطر طويلة، يمكننا جمعها في أقواس `()`
# ونجعل كل دالة في سطر منفصل.
letter_dist = (
    baby[['Last', 'Sex', 'Count']]
    .groupby(['Last', 'Sex'])
    .sum()
)
letter_dist
```

		Count
Last	**Sex**	
a	F	58079486
	M	1931630
b	F	17376
...	...	...
y	M	18569388
z	F	142023
	M	9649274

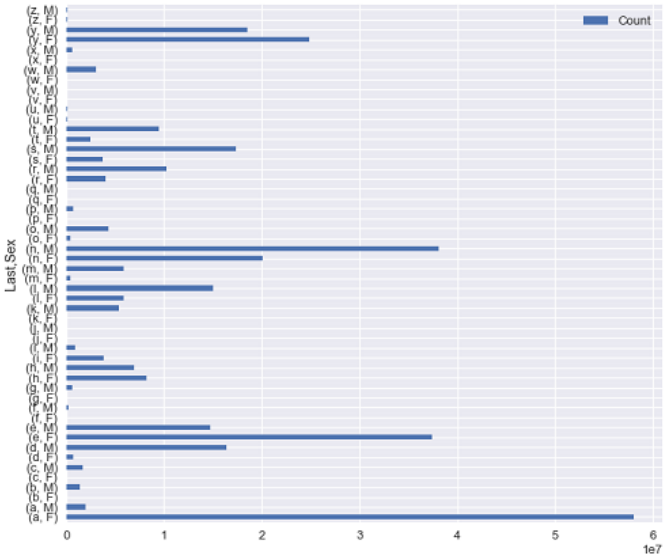
52 rows × 1 columns

الرسم البياني

تحتوي بانداز على دوال للرسم البياني والتي تحتوي على الرسوم البيانية البسيطة مثل المخطط الشريطي Bar Chart، المدرج التكراري Histogram، المخطط البياني الخطي Line Chart و مخطط التشتت Scatter plot. للرسم نستخدم الدالة .plot:

```
# نستخدم المتغير figsize لتحديد حجم الرسم البياني
letter_dist.plot.barh(figsize=(10, 10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a17af4780>
```



رغم أن الرسم البياني يوضح توزيع آخر حرف مع الجنس، يصعب التفريق بين شريط الذكور والإناث. بالرجوع لشرح الرسم البياني في مكتبة بانداز (هنا) نرى أن بانداز تقوم برسم شريط لكل سطر، وكلاهما بنفس اللون. لذا فاستخدام نسخة الجدول المحوري هنا لبياناتنا ستظهر الرسم البياني بشكل أوضح:

```
 letter_pivot = pd.pivot_table(
    baby, index='Last', columns='Sex', values='Count', aggfunc='sum'
)
letter_pivot
```

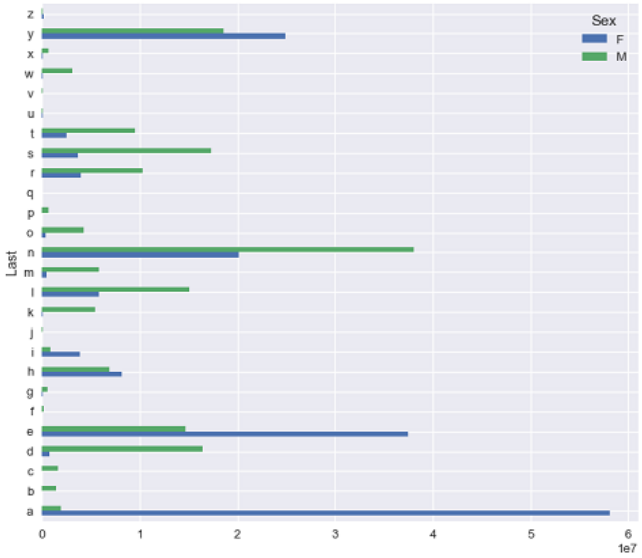
Sex	F	M
Last		
a	58079486	1931630
b	17376	1435939
c	30262	1672407
...	...	...
x	37381	644092
y	24877638	18569388
z	142023	120123

26 rows × 2 columns

استخدم الكاتب هنا pivot_table مع متغيرات مختلفة ليحدد كل قيمة بناءًا على العمود المخصص لها، الرقم الفهرسي Index حدد ليكون عمود آخر حرف Last، حدد الأعمدة هنا وهي الجنسين. حدد القيم داخل الأعمدة وهي عمود Count داخل المتغير Values، وأخيراً aggfunc وحدد فيه دالة التجميع التي نرغب باستخدامها Sum. مزيد من التفصيل هنا و هنا.

```
letter_pivot.plot.barh(figsize=(10, 10))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a17c36978>
```



قامت بانداز بتكوين مفاتيح الرسم البياني Legends لنا. ولكن لا تزال لدينا مشكلة في الرسم، أظهرنا عدد النتائج لكل حرف وجنس وفي بعض الأحيان بدأت بعض الأشرطة طويلة جداً والبعض لا يبدو واضحاً أبداً. من الأفضل أن نظهر النسب المئوية للذكور والإناث في كل حرف بدلاً من المجاميع:

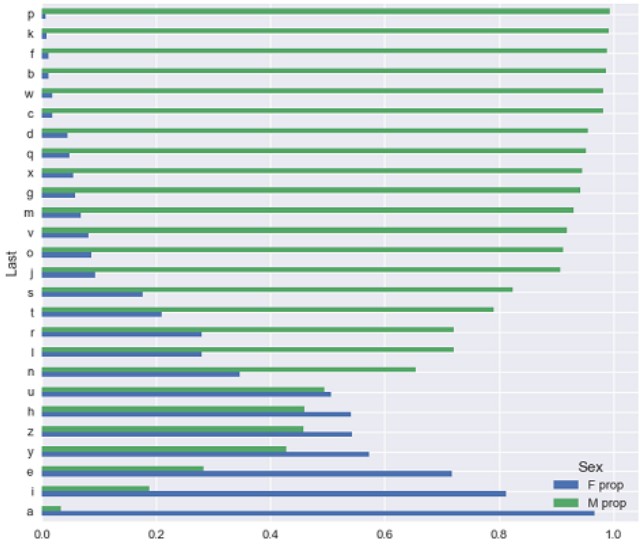
```
# قام الكاتب هنا بجمع عدد الذكور والإناث لكل حرف وجعلها في متغير جديد `total_for_each_letter`
total_for_each_letter = letter_pivot['F'] + letter_pivot['M']

# هنا قام بقسمة كل جنس من مجموع كُل حرف
letter_pivot['F prop'] = letter_pivot['F'] / total_for_each_letter
letter_pivot['M prop'] = letter_pivot['M'] / total_for_each_letter
letter_pivot
```

Sex	F	M	F prop	M prop
Last				
a	58079486	1931630	0.967812	0.032188
b	17376	1435939	0.011956	0.988044
c	30262	1672407	0.017773	0.982227
...	...	...	...	...
x	37381	644092	0.054853	0.945147
y	24877638	18569388	0.572597	0.427403
z	142023	120123	0.541771	0.458229

26 rows × 4 columns

```
(letter_pivot[['F prop', 'M prop']]
 .sort_values('M prop') # ترتيب تنازلياً حسب نسبة الذكور
 .plot.barh(figsize=(10, 10))
)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a18194b70>
```



الملخص

نلاحظ أن غالبية الأسماء المنتهية بالحرف P هم ذكور والتي تنتهي بالحرف A هم إناث! بشكل عام، مسافات الأشرطة في الرسم البياني لكثير من الأحرف تمكنا من التوقع بشكل الصحيح بجنس الشخص بناءًا على آخر حرف من أسمة.

تعلمنا استخدام الدوال التالية في باندا:

الدالة	العملية
`(series.apply(func`	استخدام دالة على جميع القيم في المجموعة
`series.str.func()`	التلاعب بالنصوص
`df.plot.func()`	الرسم البياني

التحليل الاستكشافي للبيانات

مقدمة

في التحليل الاستكشافي للبيانات، إحدى أهم الخطوات في دورة حياة علم البيانات، نقوم بالتلخيص، الرسم البياني وتحويل البيانات لشكل يسهل علينا فهمها بشكل واضح. الإحصائي "جون توكي" عرّف التحليل الاستكشافي للبيانات كالتالي:

> "التحليل الاستكشافي للبيانات" هي طريقه، حالة ذات مرونة، ورغبه في أن نرى تلك الأشياء التي نؤمن أنها ليست موجودة في البيانات، وكذلك تلك التي نؤمن بوجودها.

طالب علم البيانات قد يرى هذا التعريف غير كافي، هل طريقة الاستكشاف فقط كافيه لعمل تحليل للبيانات؟ على العكس، نقطة "توكي" هي أنه يجب علينا فهم البيانات قبل التسرع بتطبيق الاختبارات الإحصائية. يستفاد من كلام "توكي" في خوارزميات اتخاذ القرارات هذه الأيام.

في خطوة التحليل الاستكشافي للبيانات نسعى لفهم البيانات بشكل عميق. والإبقاء على "حالة المرونه" لتساعدنا على إيجاد ما نبحث عنه. فهمنا للأدوات الحسابية يساعدنا على البدء بعملية البحث. في هذا الفصل سنتعرف على الأدوات الأكثر تطوراً واستخداماً في تحليل البيانات. على الرغم من اختلاف طريقة التحليل الاستكشافي للبيانات في المجالات المختلفة، غالباً ما نبدأ الاستكشاف بفهم التالي:

- أنواع البيانات في الأعمدة وتقسيم البيانات في الصفوف.
- توزيع البيانات الرقمية ومقياس الوسط والانتشار.
- العلاقات بين البيانات الكمية.

أنواع البيانات

عادةً ما نبدأ التحليل الاستكشافي للبيانات بعرض أنواع البيانات في الجداول. على الرغم من وجود طُرق كثيره لتصنيف أنواع البيانات، في هذا الكتاب سنستخدم ثلاث من أشهر الأنواع:

- **البيانات الاسمية**: يقصد بها البيانات التي تعبر عن تصنيفات ليس لها ترتيب معين. مثلاً: الانتماء السياسي (ديموقراطي، جمهوري، غيرها)، الجنس (ذكر، انثى)، نظام الحاسب (ويندوز، ماكنتوش، لينكس).

- **البيانات الترتيبية**: بيانات بتصنيفات مرتبه. مثل: مقاسات الملابس (صغير، وسط، كبير)، مقياس ليكرت (معارض، حيادي، متفِق)، مستوى التعليم (ثانوي، بكالريوس، ماجستير). البيانات الاسمية والترتيبية تعتبر أنواع من **البيانات التصنيفية**.
- **البيانات الكمية**: وهي البيانات العددية أو الرقمية. مثل: الطول، السعر، والمسافة.

ونشير لهذه الأنواع بـ **أنواع البيانات الإحصائية**، أو للتبسيط **أنواع البيانات**.

أنواع البيانات الحسابية × الإحصائية

تقوم بانداز بتعيين نوع حسابي لكل عمود لدينا ليمثل كيف يتم حفظ البيانات في ذاكرة الحاسب.

مثلاً، لنأخذ الجدول التالي الذي يحتوي على أوزان الأطفال بعد الولادة، عرق الأم، والمستوى التعليمي للأم:

لتحميل البيانات، اضغط هنا.

```
babies_small = pd.read_csv('babies23.data', delimiter='\s+')[['wt', 'race', 'ed']]
babies_small
```

	wt	race	ed
0	120	8	5
1	113	0	5
2	128	0	2
...	...	...	...
1233	130	1	2
1234	125	0	4
1235	117	0	4

1236 rows × 3 columns

كل عمود في هذا الـ DataFrame هو حسابياً من النوع الكمي. في بياناتنا هذه، النوع `int64` يعني أن كل عمود يحتوي على عدد صحيح:

```
babies_small.dtypes
```

```
wt      int64
race    int64
ed      int64
dtype: object
```

ولكن، لن يكون مفهوماً العمل مع هذه الأعمدة كبيانات إحصائية كمية. لفهم أنواع البيانات لدينا، يجب علينا دائماً العودة إلى **قاموس البيانات**.
قاموس البيانات هو شرح تفصيلي يأتي مع البيانات يحتوي على تفاصيل وشرح أكثر لمحتوى الأعمدة. مثلاً، قاموس البيانات للبيانات التي نعمل عليها الآن هو كالتالي:

```
wt -  الوزن عند الولادة بالأونصة (999 غير معروف)
race - عرق الأم
    0-5= بيضاء
    6= مكسيكيه
    7= سوداء
    8= آسيويه
    9= منوعه
    99= غير معروف
ed - مستوى تعليم الأم
   0= اقل من السنه الثامنه,
   1= من السنة الثامنه حتى 12 - لم تتخرج,
   2= خريجة ثانويه
   3= ثانويه + مهنيه,
   4= ثانويه + جامعه
   5= خريجة جامعه
   6&7= ثانويه مهنيه غير معروفه
   9= غير معروف
```

رغم أن قيم الأعمدة الثلاثة `wt`، `race` و `ed` محفوظه كأعداد صحيحه في بانداز، العمود `race` يحتوي على بيانات أسميه والعمود `ed` يحتوي على بيانات ترتيبية.

بالأصح، يجب علينا الحذر دائماً حتى مع العمود `wt`. عند محاولة إيجاد معدل متوسط هذا العمود لن نحصل على بيانات دقيقة لأن العمود يحتوي على قيم بالرقم 999 والتي تعني قيم أوزان غير معروفة للطفل. إذا تركناه كما هو، القيم غير المعروفة ستجعل متوسط الأوزان لدينا أعلى مما يكون.

أهمية أنواع البيانات

أنواع البيانات توجهنا للطريقة الصحيحة للقيام بالعمليات على البيانات ورسمها. مثلاً، الفرق يعتبر مهم عندنا يحدث في البيانات الكمية على العكس عندما يكون في البيانات الترتيبية. هذا يعني في بيانات أوزان الأطفال بعد الولادة المتوسط للوزن يعتبر له معنى على عكس متوسط مستوى التعليم.

بانداز لن تكترث عندما نحاول إيجاد المتوسط للقيم في عمود مستوى التعليم:

```
babies_small['ed'].mean()
```

```
2.9215210355987056
```

هذه القيمة لا تقدم لنا أي معلومات. يمكن أن نستبدل القيم في العمود ed بمعانيها النصية. مثلاً، يمكن ان نستبدل 0 ب "أقل من السنة الثامنة" و 1 ب "من السنة الثامنة حتى 12 - لم تتخرج" وهكذا.

رغم أن الفرق في البيانات الترتيبية ليس له معنى، نتيجته قد تقدم لنا بعض المعلومات. مثلاً، يمكننا القول أن أم بقيمة ed=5 (خريجة جامعه) أعلى تعليماً من أم بقيمة ed=2 (خريجة ثانوية).

على العكس، لا يؤثر علينا الفرق في البيانات الاسمية. الأم التي قيمة العرق فيها race=6 (مكسيكية) و أم race=7 (سوداء) ببساطة لديهم أعراق مختلفة.

مثال: تدخين الأم وصحة الطفل الرضيع

مؤسسة صحة وتنمية الطفل (CHDS) قامت بإجراء أبحاث لفترات طويلة عن كيف أن الصحة والأمراض تنتقل بين الأجيال.

في إحدى الدراسات، قامت المؤسسة بجمع بيانات من عدة أجيال بين 1960 و 1967 لسيدات في سان فرانسيسكو - إيست باي. على الرغم أن المؤسسة تطالب بتقديم طلب للحصول على هذه البيانات، جزء من هذه البيانات تم إتاحته بواسطة قسم الإحصاء (رابط). قمنا بتحميل البيانات وقراءتها في بانداز:

لتحميل البيانات، اضغط هنا.

```
babies = pd.read_csv('babies.data', delimiter='\s+')
babies
```

	bwt	gestation	parity	age	height	weight	smoke
0	120	284	0	27	62	100	0
1	113	282	0	33	64	135	0
2	128	279	0	28	64	115	1
...	...	...	...	...	...	...	...
1233	130	291	0	30	65	150	1
1234	125	281	1	21	65	110	0
1235	117	297	0	38	65	129	0

1236 rows × 7 columns

لفهم أنواع البيانات في الأعمدة، نعود دائماً لقاموس البيانات:

العمود	الوصف
bwt	الوزن عند الولادة بالأونصة (999 غير معروف)
gestation	مدة الحمل بالأيام (999 غير معروف)
parity	هل أول مولود؟ 0=أول مولود، 9= غير معروف
age	عمر الأم بالسنوات
height	طول الأم بالبوصة (99 غير معروف)
weight	وزن الأم قبل الولادة بالباوند (999 غير معروف)
smoke	حالة تدخين الأم: 0=لا تدخن الآن، 1=تدخن الآن، 9=غير معروف

رغم أن القاموس لا يبين أنواع البيانات، يمكننا رؤية أن بعض الأعمدة تحتوي على بيانات أسميه رغم أن القيم الموجودة فيها عبارة عن أرقام. بناءًا فقط على وصف العمود يمكننا معاملة العامودين parity و smoke كبيانات أسميه والبقية كرقميه.

نتمنى دائماً ان يقدم لنا قاموس البيانات كافة المعلومات المهمة عن كل عمود، لكن من الأفضل دائماً التأكد من البيانات عن طريق فحصها. مثلاً، لا يبدو واضحاً دائماً إذا كانت القيم في عمود العمر تحتوي على أرقام (21، 30، 41) أو نطاقات (21-29، 30-40، 41+).

بيانات babies يبدو أنها تحتوي على أعداد صحيحة فقط:

```
babies
```

	bwt	gestation	parity	age	height	weight	smoke
0	120	284	0	27	62	100	0
1	113	282	0	33	64	135	0
2	128	279	0	28	64	115	1
...	...	...	...	...	...	...	...
1233	130	291	0	30	65	150	1
1234	125	281	1	21	65	110	0
1235	117	297	0	38	65	129	0

1236 rows × 7 columns

لنأخذ لمحه سريعة عن قيم عمود العمر age، يمكننا استخدام دالة series.value_counts() والتي تظهر لنا عدد مرات تكرار القيم المختلفة:

```
# تعرض العمر يسار وعدد مرات تكرارة على اليمين
babies['age'].value_counts()
```

```
23    93
26    90
24    86
      ..
45     1
44     1
15     1
Name: age, Length: 31, dtype: int64
```

يمكن أن نرى بأن الكثير من الأمهات في بياناتنا يبلغ أعمارهم بين 23 و 26 سنه. أيضاً، لا نرى أعمار بقيم كسرية (مثلاً 24.5)، يدل ذلك أن العمود age يحتوي فقط على أرقام صحيحه.

بشكل تلقائي، تقوم بانداز بتقييد عدد النتائج التي تظهر لنا. ولعرض المزيد من الأسطر يمكننا القيام بالتالي:

```
from IPython.display import display

# في هذا الحقل، اظهرنا 10 اسطر. لعرض عدد اسطر اكبر
# قم بتغير الرقم 10 في الكود البرمجي التالي إلى رقم اعلى
with pd.option_context('display.max_rows', 10):
    display(babies['age'].value_counts())
```

```
23    93
26    90
24    86
27    85
22    79
      ..
42     4
99     2
45     1
44     1
15     1
Name: age, Length: 31, dtype: int64
```

بعد القيام بمراجعة كل الأعمدة بنفس الطريقة السابقة. نقوم بتحديد نوع كل عمود بناءًا على قاموس البيانات ومراجعتنا للبيانات:

العمود	الوصف	نوع البيانات
bwt	الوزن عند الولادة بالأونصة (999 غير معروف)	كمية
gestation	مدة الحمل بالايام (999 غير معروف)	كمية
parity	هل أول مولود؟ 0=أول مولود، 9= غير معروف	أسميه
age	عمر الأم بالسنوات	كمية
height	طول الأم بالبوصة (99 غير معروف)	كمية
weight	وزن الأم قبل الولادة بالباوند (999 غير معروف)	كمية
smoke	حالة تدخين الأم: 0=لا تدخن الآن، 1=تدخن الآن، 9=غير معروف	أسميه

مثال: المخالفات الصحية في مطاعم سان فرانسيسكو

تقوم مدينة سان فرانسيسكو - كاليفورنيا دورياً بفحص المطاعم للتحقق من المخالفات الصحية. في كل عملية فحص، يحصل المطعم على تقييم من 0 إلى 100 بناءًا على عدد وأنواع المخالفات التي سجلت عليه. المطاعم، التقييم والمخالفات جميعها متوفرة بشكل عام في موقع DataSF (الرابط)، الملف يحتوي على جميع الفحوص من شهر يناير 2016.

قمنا بتحميل جزء من البيانات للمتغير scores:

لتحميل البيانات، اضغط هنا.

```
scores = pd.read_csv('SFRestaurants.csv')
scores
```

	business_name	inspection_score	violation_description	risk_category
0	All stars Donuts	86	Unclean or degraded floors walls or ceilings	Low Risk
1	Soo Fong Restaurant	92	Wiping cloths not clean or ... properly stored or	Low Risk
2	Dar Bar Pakistani/Indian Cusine	86	Moderate risk vermin infestation	Moderate Risk
...	...	...	...	...
52795	USA Power Market	71	Unclean hands or improper use of gloves	High Risk
52796	Thai Cottage Restaurant	81	Inadequate and inaccessible ...handwashing facili	Moderate Risk
52797	St Mary's Cathedral/Convention Center	90	Inadequate and inaccessible ...handwashing facili	Moderate Risk

52798 rows × 4 columns

في هذه البيانات، نحن محظوظين أن لدينا قاموس يقدم لنا معلومات أوضح بكثير عن البيانات:

العمود	نوع البيانات	الوصف
business_name	string	الاسم العام للمتجر
inspection_score	number	نتيجة الفحص، قيمة من 0 إلى 100
violation_description	string	وصف بسطر واحد للمخالفات
risk_category	string	(لم يعطى وصف)

لاحظ أن القاموس يشرح أنواع البيانات حسابياً وليس أنواعها بشكل إحصائي. مثلاً، يستخدم `string` بدلاً من التحديد إذا كانت أسميه أو ترتيبية. ولكن نلاحظ أن العمود `risk_category` لا يحتوي على وصف.

البيانات النصية ليست دائماً أسميه

العمود `violation_description` يبين لنا وصف للمخالفات. لكن لم يوضح لنا في القاموس إذا ما علينا معاملة البيانات في هذا العمود كبيانات أسميه. قد يحتوي العمود على بيانات **نصيه غير منظمة**، أو بيانات أدخلت يدوياً. النصوص غير المنظمة عاده ما تأتي من مصادر بيانات بلغه طبيعية، مثل مقالات الصحف، بحث قوقل، و ردود أسئلة الاستبيانات.

عاده لا نتعامل مع تلك البيانات كبيانات أسميه، البيانات الاسمية عاده تتكون من تصنيفات محددة مسبقاً، على عكس النصوص الغير منظمة. لذا، يجيب علينا التحقق من القيم في العمود `violation_description` لتحديد ما اذا كانت تحتوي على بيانات نصيه غير منظمة أو بيانات أسميه.

إذا كان العمود يحتوي على بيانات نصيه غير منظمة، فيجب أن نتوقع تكرار قليل لبعض القيم بسبب وجود طرق كثيره لتسجيل نفس المخالفة، مثلاً: `unclean floors` "أرضيه غير نظيفة", `dirty floors` "أرضيه قذرة", `floor needs cleaning` "الأرضية تحتاج للتنظيف" وغيرها الكثير. في الجانب الآخر، إذا كان العمود يحتوي على قيم محددة مسبقاً، الكثير منها سيتكرر.

في المثال السابق على البيانات النصية الغير منظمة يوضح الكاتب طرق مختلفة لكتابة مخالفات لها نفس المعنى، يعني هنا أمثله لتكرار نفس المخالفات لكن كُتبت بأكثر من طريقه. لذلك التكرار فيها سيكون أقل بكثير من البيانات الاسمية. في حال أردنا أن تكون هذه البيانات أسميه، فيجب علينا تحديد قيم مسبقة مثلاً `unclean floors` وتستخدم بدلاً من القيم الثلاثة.

نريد أن نتحقق ما أذا كان العمود يحتوي على قيم مكررة:

```
with pd.option_context('display.max_rows', 14):
    display(scores['violation_description'].value_counts())
```

```
Unclean or degraded floors walls or ceilings                          3668
Unapproved or unmaintained equipment or utensils                      2704
Inadequate and inaccessible handwashing facilities                    2653
Moderate risk food holding temperature                                2588
Inadequately cleaned or sanitized food contact surfaces               2467
Wiping cloths not clean or properly stored or inadequate sanitizer    2121
Foods not protected from contamination                                1929
                                                                      ...
Discharge from employee nose mouth or eye                                6
Noncompliance with Gulf Coast oyster regulation                          5
Mobile food facility stored in unapproved location                       4
Mobile food facility with unapproved operating conditions                3
Unreported or unrestricted ill employee with communicable disease        1
Noncompliance with Cottage Food Operation                                1
Mobile food facility HCD insignia unavailable                            1
Name: violation_description, Length: 67, dtype: int64
```

النتيجة جعلتنا نتأكد أن القيم المحفوظة في هذا العمود تم تحديدها مسبقاً في قائمة للمخالفات المُحتملة. لذا، سنتعامل مع العمود `violation_description` كعمود يحتوي على بيانات أسميه.

التحقق من الوصف المفقود

رغم أن العمود `risk_category` لم يتم وصفه في قاموس البيانات، يمكننا التحقق من محتوى العمود لمحاولة فهم معناه. أولاً، نلاحظ أن العمود يحتوي على ثلاث قيم فقط:

```
scores['risk_category'].value_counts()
```

```
Low Risk         19694
Moderate Risk    14712
High Risk         5686
Name: risk_category, dtype: int64
```

يمكن أن نفهم أن هذه القيم تشرح خطورة المخالفة. يمكننا التحقق من فهمنا عن طريق عرض المخالفات لكل قيمه من القيم الموجودة في العمود `risk_category`:

```
def risk_counts(risk):
    return (scores.loc[scores['risk_category'] == risk,
                       'violation_description']
            .value_counts().head())
```

الكاتب عرّف دالة تستقبل نص وهنا النص إحدى الأنواع الثلاثة من درجات خطورة المخالفة، وينتج عن الدالة DataFrame يحتوي على كل نوع من المخالفات في تلك الدرجة وعدد مرات تكرارها. يمكنك مراجعة الفصل السابق لمعلومات أكثر عن `.loc` وكيفية إيجاد الأسطر التي تحتوي على قيم معينة.

```
risk_counts('High Risk')
```

```
High risk food holding temperature             1619
Unclean or unsanitary food contact surfaces    1197
Improper cooling methods                        823
Unclean hands or improper use of gloves         755
High risk vermin infestation                    712
Name: violation_description, dtype: int64
```

```
risk_counts('Moderate Risk')
```

```
Inadequate and inaccessible handwashing facilities         2653
Moderate risk food holding temperature                     2588
Inadequately cleaned or sanitized food contact surfaces    2467
Foods not protected from contamination                     1929
Moderate risk vermin infestation                           1814
Name: violation_description, dtype: int64
```

```
risk_counts('Low Risk')
```

```
Unclean or degraded floors walls or ceilings                          3668
Unapproved or unmaintained equipment or utensils                      2704
Wiping cloths not clean or properly stored or inadequate sanitizer    2121
Improper food storage                                                 1817
Unclean nonfood contact surfaces                                      1440
Name: violation_description, dtype: int64
```

وبشكل سريع نلاحظ، أن أنواع المخالفات مقسمه حسب درجة الخطورة. أيضاً، المخالفات من الدرجة `High Risk` احتمالية تسببها لأمراض أكثر من المخالفات من الدرجة `Low Risk`. من المحتمل أن يكون مطعماً بمخالفة `High risk vermin infestation` "خطورة حصول غزو/هجوم حشري" أقل نظافة و احتماليه تعرض مرتاديه لأمراض أكثر من مطعم بمخالفة `Unclean nonfood contact surfaces` "أسطح غير مخصصه للطعام غير نظيفة".

بسبب ذلك، قررنا أن العمود `risk_category` يحتوي على بيانات ترتيبية توضح درجة خطورة كل مخالفه (Low Risk < Moderate Risk < High Risk).

مراجعة قاموس البيانات

العمود	نوع البيانات	الوصف
business_name	string	الاسم العام للمتجر
inspection_score	number	نتيجة الفحص، قيمة من 0 إلى 100
violation_description	string	وصف بسطر واحد للمخالفات
risk_category	string	درجة خطورة المخالفة (Low Risk < Moderate Risk < High Risk)

ملخص

عرفنا البيانات الاسمية، الترتيبية والكمية وأهميتها في علم البيانات. شاهدنا أمثله على بيانات، واستعنا بقواميس البيانات وبالبيانات نفسها لتحديد أنواع البيانات في كل عمود. تأكد أن تفرق بين أنواع البيانات الحسابية والإحصائية.

تنظيف البيانات

مقدمة

تأتي البيانات بعدة أشكال وتتنوع من حيث فائدتها في التحليل. رغم أننا نسعى أن تكون جميع البيانات على شكل جدول وكل قيمه تم إدخالها بشكل صحيح ودقيق، ولكن يجب علينا التحقق بشكل دقيق عن المشاكل التي قد تقودنا لنتائج غير صحيحه.

المصطلح **تنظيف البيانات** يطلق على خطوة التحقق من البيانات واتخاذ قرارات عن كيفية إصلاح الأخطاء وتعويض البيانات المفقودة. سنناقش أكثر المشاكل الشائعة في البيانات ونشرحها بشكل أوضح.

تنظيف البيانات له بعض القيود. مثلاً، لا يوجد تنظيف بإمكانه تحسين عينات تم أخذها بشكل متحيز. قبل البدء بمشوار تنظيف البيانات الذي عاده ما يكون طويلاً، يجب ان نتأكد ان بياناتنا تم جمعها بشكل دقيق ودون تحيز. في ذلك الحين يمكننا بدأ التحقق من البيانات وتنظيفها للتخلص من المشاكل في أنواع البيانات أو طرق الإدخال.

سنقوم بشرح طُرق لتنظيف البيانات وسنستخدم بيانات شرطة مدينة بيركلي.

نظره على بيانات شرطة مدينة بيركلي

سنستخدم بيانات شرطة مدينة بيركلي المنشورة للعامة لشرح طرق تنظيف البيانات. يمكن تحميل بيانات المكالمات من هنا، وبيانات الإيقافات من هنا.

روابط أخرى لتحميل البيانات:

- بيانات المكالمات.
- بيانات الإيقافات.

سنستخدم الأمر ls مع -lh لعرض معلومات أكثر عن الملفات:

```
!ls -lh data/
```

```
total 13936
-rw-r--r--@ 1 sam  staff   979K Aug 29 14:41 Berkeley_PD_-_Calls_for_Service.csv
-rw-r--r--@ 1 sam  staff    81B Aug 29 14:28 cvdow.csv
-rw-r--r--@ 1 sam  staff   5.8M Aug 29 14:41 stops.json
```

يظهر لنا الأمر السابق الملفات في المجلد وأحجامها. الأمر مهم لأنه يبين لنا أن الملفات صغيرة الحجم ويمكن تحميلها على ذاكرة حاسبنا. قاعدة عامه، يمكن تحميل الملفات بشكل آمن عندما يكون حجمها حوالي ربع حجم الذاكرة في الحاسب. مثلاً، إذا كان جهازنا يحتوي على 4 ج.ب. من الذاكرة العشوائية، يمكننا تحميل ملف CSV بحجم 1 ج.ب في بانداز. ليتحمل حاسبنا ملفات بحجم أكبر يجب علينا استخدام أدوات خاصة وسنتحدث لاحقاً عنها في هذا الكتاب.

لاحظ استخدامنا لعلامة التعجب قبل ls. هذا يخبر جوبتر أن الكود البرمجي التالي خاص بمترجم الشيل، وليس بايثون. مثال آخر:

```
# الامر `wc` يظهر عدد الاسطر في كل ملف.
# يمكن ان نلاحظ ان ملف `stops.json` سطر (29852) يحتوي على عدد.
!wc -l data/*
```

```
   16497 data/Berkeley_PD_-_Calls_for_Service.csv
       8 data/cvdow.csv
   29852 data/stops.json
   46357 total
```

تنظيف بيانات المكالمات

فهم توليد البيانات

سنقوم بطرح بعض الأسئلة التي يجب أن تتحقق منها في جميع بياناتك قبل البدء بتنظيفها ومعالجتها. هذه الأسئلة عن كيف تم توليد وإنشاء هذه البيانات. في هذه الخطوة، تنظيف البيانات **لن** يساعدنا على حل المشاكل التي حصلت أثناء إنشاء البيانات.

على ماذا تحتوي البيانات؟ الموقع الذي تم أخذ بيانات المكالمات منه يذكر أن "الجرائم/الحوادث (وليس تقارير الجرائم) التي حدثت في ال180 يوم السابقة". مزيد من القراءة في الموقع أوضحت التالي "ليس جميع المكالمات التي طلبت خدمات الشرطة تم إضافتها (مثال لأحد الحالات التي تمت ولم يتم إضافتها: حالة عض حيوان لشخص)".

موقع بيانات الإيقافات يوضح أن البيانات تحتوي على جميع "السيارات المحتجزة (بما فيها الدراجات الهوائية) واعتقالات المشاة (بحد أعلى خمسة أشخاص)" منذ 26 يناير 2015.

هل البيانات تعداد / إحصاء على السكان؟ يعتمد ذلك على هدفنا من تحليل البيانات. مثلاً، إذا كنا مهتمين بمعرفة المكالمات التي طلبت فيها الخدمات لآخر 180 يوم للحوادث والجرائم، فإن بيانات المكالمات تعتبر بيانات إحصائية للمكالمات التي تمت آخر 180 يوم من قبل السكان. ولكن إذا أردنا المكالمات لطلب خدمات الشرطة للعشر سنوات السابقة، فإن البيانات ليست مناسبة كونها تحتوي على بيانات آخر 180 يوم فقط. يمكننا ذكر نفس الكلام على بيانات الإيقافات لأن البيانات تم جمعها ابتداء من 26 يناير 2015.

إذا كانت البيانات تُشكل عينه، هل هي عينة الاحتمالات؟ البيانات لا تمثل عينة الاحتمالات لأنها لا تظهر أي عشوائية في طريقة جمع البيانات. لدينا بيانات كاملة لفترة معينة فقط وليس لفترات أخرى.

هل هناك قيود ستجبرنا عليه البيانات في نتائجنا؟ رغم أننا سنسأل هذا السؤال بعد كل خطوه من خطواتنا، يمكننا ملاحظة أن بياناتنا تظهر لنا بعض القيود. أهم القيود التي تظهرها لنا هي أننا لا يمكننا القيام بأي تقديرات غير متحيزة للفترات التي لم يتم تسجيلها في البيانات.

تنظيف البيانات

لنبدأ الآن بتنظيف بيانات المكالمات. الأمر head يظهر لنا أول خمس أسطر في الملف:

```
!head data/Berkeley_PD_-_Calls_for_Service.csv
```

```
CASENO,OFFENSE,EVENTDT,EVENTTM,CVLEGEND,CVDOW,InDbDate,Block_Location,BLKADDR,City,State
17091420,BURGLARY AUTO,07/23/2017 12:00:00 AM,06:00,BURGLARY - VEHICLE,0,08/29/2017 08:28:05 AM,"2500 LE C
Berkeley, CA
(37.876965, -122.260544)",2500 LE CONTE AVE,Berkeley,CA
17020462,THEFT FROM PERSON,04/13/2017 12:00:00 AM,08:45,LARCENY,4,08/29/2017 08:28:00 AM,"2200 SHATTUCK AV
Berkeley, CA
(37.869363, -122.268028)",2200 SHATTUCK AVE,Berkeley,CA
17050275,BURGLARY AUTO,08/24/2017 12:00:00 AM,18:30,BURGLARY - VEHICLE,4,08/29/2017 08:28:06 AM,"200 UNIVE
```

```
Berkeley, CA
(37.865491, -122.310065)",200 UNIVERSITY AVE,Berkeley,CA
```

يظهر أن الملف من النوع CSV، ولكن من الصعب معرفة ما إذا كان جميع محتوى الملف منسق بطريقه صحيحه. يمكننا استخدام الدالة pd.read_csv لقراءة الملف ك DataFrame. إذا أظهر الأمر pd.read_csv أخطاء، فيجب علينا البحث بشكل أعمق وحل المشكلة يدوياً. لحسن الحظ، قامت الدالة بقراءة الملف بشكل صحيح على شكل DataFrame:

```
import pandas as pd

calls = pd.read_csv('data/Berkeley_PD_-_Calls_for_Service.csv')
calls
```

FFENSE	EVENTDT	EVENTTM	...	Block_Location	BLKADDR	City	State
IRGLARY AUTO	07/23/2017 12:00:00 AM	06:00	...	LE CONTE 2500 AVE\nBerkeley, ,CA\n(37.876965 ...-	LE 2500 CONTE AVE	Berkeley	CA
:FT FROM ERSON	04/13/2017 12:00:00 AM	08:45	...	SHATTUCK 2200 AVE\nBerkeley, ,CA\n(37.869363 ...-	2200 SHATTUCK AVE	Berkeley	CA
IRGLARY AUTO	08/24/2017 12:00:00 AM	18:30	...	UNIVERSITY 200 AVE\nBerkeley, ,CA\n(37.865491 ...	200 UNIVERSITY AVE	Berkeley	CA
...	...	...	...	...	...	...	...
'URBANCE	04/01/2017 12:00:00	12:22	...	FAIRVIEW 1600 ST\nBerkeley, CA\n(37.850001, ...-1	1600 FAIRVIEW ST	Berkeley	CA
:FT MISD. IDER $950	04/01/2017 12:00:00	12:00	...	DELAWARE 2000 ST\nBerkeley, CA\n(37.874489, ...-1	2000 DELAWARE ST	Berkeley	CA
:EXUAL SSAULT .MISD	08/22/2017 12:00:00 AM	20:02	...	2400 TELEGRAPH AVE\nBerkeley, ,CA\n(37.866761 ...	2400 TELEGRAPH AVE	Berkeley	CA

5508 rows × 11 columns

يمكننا كتابة دالة تقوم بإظهار أجزاء من البيانات:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual

def df_interact(df):
    def peek(row=0, col=0):
        return df.iloc[row:row + 5, col:col + 6]
    interact(peek, row=(0, len(df), 5), col=(0, len(df.columns) - 6))
    print('({} rows, {} columns) total'.format(df.shape[0], df.shape[1]))

df_interact(calls)
```

الكود البرمجي السابق يُنتج لنا جدول تفاعلي كما في الصورة أدناه، يحتوي بشكل تلقائي على ال6 أسطر الأولى و أول 6 أعمدة. عند تحريك أي من الشريطين تظهر لنا أسطر / أعمدة مختلفة بناءًا على رقم السطر الذي اخترناه ورقم الأعمدة. تذكر دائماً أنه الترقيم في الجداول / بايثون يبدأ من 0.

row 0

col 0

	CASENO	OFFENSE	EVENTDT	EVENTTM	CVLEGEND	CVDOW
0	17091420	BURGLARY AUTO	07/23/2017 12:00:00 AM	06:00	BURGLARY - VEHICLE	0
1	17020462	THEFT FROM PERSON	04/13/2017 12:00:00 AM	08:45	LARCENY	4
2	17050275	BURGLARY AUTO	08/24/2017 12:00:00 AM	18:30	BURGLARY - VEHICLE	4
3	17019145	GUN/WEAPON	04/06/2017 12:00:00 AM	17:30	WEAPONS OFFENSE	4
4	17044993	VEHICLE STOLEN	08/01/2017 12:00:00 AM	18:00	MOTOR VEHICLE THEFT	2

```
(5508 rows, 11 columns) total
```

بناءًا على النتائج في الأعلى، تبدو البيانات مرتبه بشكل مناسب بما ان الأعمدة مُسماه بشكل صحيح والبيانات في كل عمود مدخله بشكل متناسق. ماذا يحتوي كل عمود؟ يمكننا التحقق من ذلك بموقع البيانات:

العمود	الوصف	النوع
CASENO	رقم القضية	رقم
OFFENSE	نوع المخالفة	نص
EVENTDT	تاريخ الحدث	تاريخ + وقت
EVENTTM	وقت الحدث	نص
CVLEGEND	تفاصيل الحدث	نص
CVDOW	أي يوم بالأسبوع حدثت فيه المخالفة	رقم
InDbDate	تاريخ ووقت تحديث المخالفة في قاعدة البيانات	تاريخ + وقت
Block_Location	عنوان المخالفه	عنوان
BLKADDR		نص
City		نص
State		نص

قد تبدو البيانات سهلة التحليل. ولكن قبل البدء بذلك، يجب أن نجيب عن الأسئلة التالية:

- **هل توجد بيانات مفقوده؟** السؤال مهم لأن البيانات المفقودة قد تكون لعدة أسباب. مثلاً، عناوين مفقوده قد يكون حذفها بسبب حماية خصوصية الأشخاص، أو أن أحد المباشرين لهذه المخالفة رفض الإجابة على هذا السؤال، أو بسبب حدوث مشاكل في جهاز التسجيل.
- **هل توجد أي بيانات مفقوده تم تعبئتها؟ (مثلاً كتابة رقم 999 لعمر مجهول أو 12:00 صباحاً لتاريخ مجهول)؟** بالتأكيد سيأثر ذلك على تحليلنا إذا تجاهلناها.
- **أي جزء من البيانات أدخلت بواسطة أشخاص حقيقيين؟** كما سنرى بعد قليل، البيانات التي يتم ادخالها من البشر تحتوي على الكثير من التناقضات والأخطاء الإملائيه.

رغم أن هناك الكثير من المشاكل التي يجب أن نتحقق منها، هذه المشاكل الثلاثة هي الأكثر تكراراً في جميع البيانات. يمكنك مراجعة Quartz لاستعراض قائمة من المشاكل التي يحتاج للتحقق منها في البيانات قبل البدء بالتحليل.

هل توجد بيانات مفقوده؟

هذه الخطوة سهله التحقق في باندا‌ز عن طريق الكود البرمجي التالي:

```
ستظهر لنا الأسطر التي تحتوي على الاقل على قيمه واحده مفقوده #
null_rows = calls.isnull().any(axis=1)
calls[null_rows]
```

OFFENSE	EVENTDT	EVENTTM	...	Block_Location	BLKADDR	City	State
BURGLARY AUTO	03/16/2017 12:00:00 AM	22:00	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA
BURGLARY AUTO	07/20/2017 12:00:00 AM	16:00	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA
VEHICLE STOLEN	04/22/2017 12:00:00 AM	21:00	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA
...	...	...	...	...	...	...	...
VANDALISM	07/01/2017 12:00:00	08:00	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA

OFFENSE	EVENTDT	EVENTTM	...	Block_Location	BLKADDR	City	State
BURGLARY RESIDENTIAL	06/30/2017 12:00:00 AM	15:00	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA
VANDALISM	08/15/2017 12:00:00 AM	23:30	...	Berkeley, CA\n(37.869058, (-122.270455	NaN	Berkeley	CA

27 rows × 11 columns

يظهر لنا 27 سطرًا لا تحتوي على عناوين في العمود BLKADDR. لسوء الحظ، لا يظهر لنا في شرح البيانات اي معلومه عن طريقة حفظ معلومات العنوان. نعرف أن جميع البيانات لأحداث تمت في مدينة بيركلي، لذا يمكننا الافتراض ان جميع المكالمات كانت لعناوين في مكان ما في بيركلي.

هل توجد أي بيانات مفقوده تم تعبئتها؟

من النتيجه السابقه نلاحظ ان العمود Block_Location يحتوي على القيمه Berkeley, CA اذا كان القيمه في العمود BLKADDR مفقوده.

أيضاً، التحقق من الجدول اظهر لنا ان العمود EVENTDT يحتوي على التاريخ بشكل صحيح ولكن في كل الأسطر تم تسجيل الوقت 12:00:00AM ، الوقت الحقيقي في العمود EVENTTM. لنتحقق من ما اكتشفناه:

```
# اظهار اول سبع اسطر
calls.head(7)
```

FFENSE	EVENTDT	EVENTTM	...	Block_Location	BLKADDR	City	State
RGLARY AUTO	07/23/2017 12:00:00 AM	06:00	...	LE CONTE 2500 AVE\nBerkeley, ,CA\n(37.876965 ...-	LE 2500 CONTE AVE	Berkeley	CA
FT FROM ERSON	04/13/2017 12:00:00 AM	08:45	...	SHATTUCK 2200 AVE\nBerkeley, ,CA\n(37.869363 ...-	2200 SHATTUCK AVE	Berkeley	CA
RGLARY AUTO	08/24/2017 12:00:00 AM	18:30	...	UNIVERSITY 200 AVE\nBerkeley, ,CA\n(37.865491 ...	200 UNIVERSITY AVE	Berkeley	CA
/WEAPON	04/06/2017 0:00	17:30	...	SEVENTH 1900 ST\nBerkeley, CA\n(37.869318, ...-12	1900 SEVENTH ST	Berkeley	CA
EHICLE TOLEN	08/01/2017 0:00	18:00	...	PARKSIDE 100 DR\nBerkeley, CA\n(37.854247, ...-12	100 PARKSIDE DR	Berkeley	CA
RGLARY IDENTIAL	06/28/2017 12:00:00 AM	12:00	...	PRINCE 1500 ST\nBerkeley, CA\n(37.851503, ...-122	1500 PRINCE ST	Berkeley	CA
RGLARY IDENTIAL	05/30/2017 12:00:00 AM	08:45	...	MENLO 300 PL\nBerkeley, CA\n	MENLO 300 PL	Berkeley	CA

7 rows × 11 columns

كخطوة تنظيف، نريد أن نجمع الأعمدة EVENTDT و EVENTTM ليحتوي على التاريخ والوقت في عمود واحد. إذا قمنا بكتابة دالة تستقبل DataFrame وتنشأ أخرى، فيمكننا لاحقنا استخدام pd.pipe لتطبيق ذلك على جميع القيم:

```
def combine_event_datetimes(calls):
    combined = pd.to_datetime(
        # جمع التاريخ والوقت على شكل نص
        calls['EVENTDT'].str[:10] + ' ' + calls['EVENTTM'],
        infer_datetime_format=True,
    )
    return calls.assign(EVENTDTTM=combined)

# لعرض النتائج قبل التعديل على ال DataFrame
calls.pipe(combine_event_datetimes).head(2)
```

	OFFENSE	EVENTDT	EVENTTM	...	BLKADDR	City	State	EVENTDTTM
	BURGLARY AUTO	07/23/2017 12:00:00 AM	06:00	...	LE 2500 CONTE AVE	Berkeley	CA	23/07/2017 06:00:00
	THEFT FROM PERSON	04/13/2017 12:00:00 AM	08:45	...	2200 SHATTUCK AVE	Berkeley	CA	2017-04-13 08:45:00

2 rows × 12 columns

أي جزء من البيانات أدخلت بواسطة أشخاص حقيقيين؟

يبدو أن الكثير من الأعمدة تم إدخالها بشكل تلقائي بواسطة الآلة، بما في ذلك التاريخ، الوقت، اليوم في الأسبوع، وعنوان الحادثة.

أيضاً، الأعمدة `OFFENSE` و `CVLEGEND` يبدو أنها تحتوي على بيانات ثابتة. يمكننا التحقق من القيم المدخلة في كل عمود لنتحقق إن كان هناك أي قيم تحتوي على أخطاء إملائية:

```
calls['OFFENSE'].unique()
```

```
array(['BURGLARY AUTO', 'THEFT FROM PERSON', 'GUN/WEAPON',
       'VEHICLE STOLEN', 'BURGLARY RESIDENTIAL', 'VANDALISM',
       'DISTURBANCE', 'THEFT MISD. (UNDER $950)', 'THEFT FROM AUTO',
       'DOMESTIC VIOLENCE', 'THEFT FELONY (OVER $950)', 'ALCOHOL OFFENSE',
       'MISSING JUVENILE', 'ROBBERY', 'IDENTITY THEFT',
       'ASSAULT/BATTERY MISD.', '2ND RESPONSE', 'BRANDISHING',
       'MISSING ADULT', 'NARCOTICS', 'FRAUD/FORGERY',
       'ASSAULT/BATTERY FEL.', 'BURGLARY COMMERCIAL', 'MUNICIPAL CODE',
       'ARSON', 'SEXUAL ASSAULT FEL.', 'VEHICLE RECOVERED',
       'SEXUAL ASSAULT MISD.', 'KIDNAPPING', 'VICE', 'HOMICIDE'], dtype=object)
```

```
calls['CVLEGEND'].unique()
```

```
array(['BURGLARY - VEHICLE', 'LARCENY', 'WEAPONS OFFENSE',
       'MOTOR VEHICLE THEFT', 'BURGLARY - RESIDENTIAL', 'VANDALISM',
       'DISORDERLY CONDUCT', 'LARCENY - FROM VEHICLE', 'FAMILY OFFENSE',
       'LIQUOR LAW VIOLATION', 'MISSING PERSON', 'ROBBERY', 'FRAUD',
       'ASSAULT', 'NOISE VIOLATION', 'DRUG VIOLATION',
       'BURGLARY - COMMERCIAL', 'ALL OTHER OFFENSES', 'ARSON', 'SEX CRIME',
       'RECOVERED VEHICLE', 'KIDNAPPING', 'HOMICIDE'], dtype=object)
```

بما أن كل قيمه يبدو أنها أدخلت بشكل صحيح، لن نحتاج للقيام بأي تعديلات على العمودان.

تحققنا أيضاً من العمود `BLKADDR` إن كان يحتوي على أي تناقضات ووجدنا أن العناوين أدخلت في بعض المرات على الشكل التالي `LE CONTE 2500 AVE` كعنوان كامل وفي بعض مرات أخرى سجل العنوان كتقاطع شارعين مثل `ALLSTON WAY & FIFTH ST`. يشير ذلك أن البيانات أدخلت يدوياً بواسطة أشخاص حقيقيين وليس بشكل آلي، هذا العمود سيكون صعب تحليله. لحسن الحظ يمكننا استخدام بيانات خطوط الطول والعرض `Latitude` و `Longitude` بدلاً من العنوان.

```
calls['BLKADDR'][[0, 5001]]
```

```
0           2500 LE CONTE AVE
5001    ALLSTON WAY & FIFTH ST
Name: BLKADDR, dtype: object
```

لمسات أخيره

تبدو البيانات جاهزة للتحليل الآن. العمود `Block_Location` يحتوي على نص بالعنوان، خط الطول والعرض. نحتاج لفصل قيمة خط الطول والعرض ليسهل استخدامها:

```
def split_lat_lon(calls):
    return calls.join(
        calls['Block_Location']
```

```
        # الحصول على الإحداثات من النص ( بيانات خط الطول والعرض)
        .str.split('\n').str[2]
        # حذف الأقواس من النص
        .str[1:-1]
        # فصل قيم خط الطول والعرض إلى عمودان
        .str.split(', ', expand=True)
        .rename(columns={0: 'Latitude', 1: 'Longitude'})
    )

calls.pipe(split_lat_lon).head(2)
```

OFFENSE	EVENTDT	EVENTTM	CVDOW	...	City	State	Latitude	Longitude
BURGLARY AUTO	07/23/2017 12:00:00 AM	6:00	0	...	Berkeley	CA	37.876965	122.260544-
THEFT FROM PERSON	04/13/2017 12:00:00 AM	8:45	0	...	Berkeley	CA	37.869363	122.268028-

2 rows × 13 columns

ثم نربط كل رقم في العمود CVDOW مع نص اليوم، سنستخدم الملف cvdow.csv الذي تم تجهيزه مسبقاً، يحتوي الملف على رقم اليوم واليوم كنص مكتوب:

لتحميل الملف: cvdow.csv

```
day_of_week = pd.read_csv('data/cvdow.csv')
day_of_week
```

	CVDOW	Day
0	0	Sunday
1	1	Monday
2	2	Tuesday
3	3	Wednesday
4	4	Thursday
5	5	Friday
6	6	Saturday

نقوم بترجمة العمود CVDOW إلى نص:

```
def match_weekday(calls):
    return calls.merge(day_of_week, on='CVDOW')

calls.pipe(match_weekday).head(2)
```

IO	OFFENSE	EVENTDT	EVENTTM	...	BLKADDR	City	State	Day
420	BURGLARY AUTO	07/23/2017 12:00:00 AM	6:00	...	LE 2500 CONTE AVE	Berkeley	CA	Sunday
302	BURGLARY AUTO	07/02/2017 0:00	22:00	...	BOWDITCH STREET & CHANNING WAY	Berkeley	CA	Sunday

2 rows × 12 columns

سنحذف الأعمدة التي لا نحتاجها:

```
def drop_unneeded_cols(calls):
    return calls.drop(columns=['CVDOW', 'InDbDate', 'Block_Location', 'City',
                               'State', 'EVENTDT', 'EVENTTM'])
```

الآن نقوم بتطبيق جميع الدوال التي عرفناها سابقاً على البيانات باستخدام pipe:

```
calls_final = (calls.pipe(combine_event_datetimes)
               .pipe(split_lat_lon)
               .pipe(match_weekday)
               .pipe(drop_unneeded_cols))
df_interact(calls_final)
```

بنفس الطريقة السابقة سيظهر لنا جدول تفاعلي.

row 0

col 0

	CASENO	OFFENSE	CVLEGEND	BLKADDR	EVENTDTTM	Latitude
0	17091420	BURGLARY AUTO	BURGLARY - VEHICLE	2500 LE CONTE AVE	2017-07-23 06:00:00	37.876965
1	17038302	BURGLARY AUTO	BURGLARY - VEHICLE	BOWDITCH STREET & CHANNING WAY	2017-07-02 22:00:00	37.867209
2	17049346	THEFT MISD. (UNDER $950)	LARCENY	2900 CHANNING WAY	2017-08-20 23:20:00	37.867948
3	17091319	THEFT MISD. (UNDER $950)	LARCENY	2100 RUSSELL ST	2017-07-09 04:15:00	37.856719
4	17044238	DISTURBANCE	DISORDERLY CONDUCT	TELEGRAPH AVENUE & DURANT AVE	2017-07-30 01:16:00	37.867816

(5508 rows, 8 columns) total

الآن، بيانات المكالمات جاهزة للتحليل. في الجزء التالي، سنقوم بتنظيف بيانات الإيقافات.

```
calls_final.to_csv('data/calls.csv', index=False)
```

تنظيف بيانات الإيقافات

لنبدأ تجهيز بيانات ملف الإيقافات للتحليل.

لتحميل الملف: stops.json

نستخدم head لعرض الأسطر الأولى في الملف:

```
!head data/stops.json
```

```
{
  "meta" : {
    "view" : {
      "id" : "6e9j-pj9p",
      "name" : "Berkeley PD - Stop Data",
      "attribution" : "Berkeley Police Department",
      "averageRating" : 0,
      "category" : "Public Safety",
      "createdAt" : 1444171604,
      "description" : "This data was extracted from the Department’s Public Safety Server and covers the c
```

كما يبدو واضحاً أن الملف ليس من النوع CSV. الملف يحتوي على بيانات من النوع JSON (JavaScript Object Notation) "ترميز الكائنات باستعمال جافا سكريبت"، طريقه كثيرة الاستخدام تُحفظ فيها البيانات على في مصفوفة من النوع Dictionary. مكتبة JSON في بايثون تسهل علينا قراءة هذا النوع من المصفوفات:

تم شرح هذا النوع من المصفوفات Dictionary في الفصل الثالث. تحتوي على مفاتيح Keys وقيم لها Values.

```
import json

# انتبه ان الملف قد يستهلك جميع ذاكرة الجهاز اذا كان حجمه كبير
# تحققا من الحجم قبل تحميل الملف

with open('data/stops.json') as f:
    stops_dict = json.load(f)
```

```
stops_dict.keys()
```

```
dict_keys(['meta', 'data'])
```

لاحظ أننا أظهرنا فقط المفاتيح في المصفوفة stops_dict كي لا نتسبب ببطيء المتصفح أثناء طباعة جميع محتوى المصفوفة. لإلقاء نظره على البيانات دون أن نتسبب بتعطيل المتصفح بسبب حجم البيانات الكبير، يمكننا تحويل المصفوفة إلى نص وطباعة بعض من محتواها:

```
from pprint import pformat

def print_dict(dictionary, num_chars=1000):
    print(pformat(dictionary)[:num_chars])

print_dict(stops_dict['meta'])
```

عرف الكاتب هنا دالة لطباعة محتوى المصفوفة، تستقبل الدالة المصفوفة + عدد الأحرف التي يرغب بطباعتها

```
{'view': {'attribution': 'Berkeley Police Department',
          'averageRating': 0,
          'category': 'Public Safety',
          'columns': [{'dataTypeName': 'meta_data',
                       'fieldName': ':sid',
                       'flags': ['hidden'],
                       'format': {},
                       'id': -1,
                       'name': 'sid',
                       'position': 0,
                       'renderTypeName': 'meta_data'},
                      {'dataTypeName': 'meta_data',
                       'fieldName': ':id',
                       'flags': ['hidden'],
                       'format': {},
                       'id': -1,
                       'name': 'id',
                       'position': 0,
                       'renderTypeName': 'meta_data'},
                      {'dataTypeName': 'meta_data',
                       'fieldName': ':position',
                       'flags': ['hidden'],
                       'format': {},
```

```
print_dict(stops_dict['data'], num_chars=300)
```

```
[[1,
  '29A1B912-A0A9-4431-ADC9-FB375809C32E',
  1,
  1444146408,
  '932858',
  1444146408,
  '932858',
  None,
  '2015-00004825',
  '2015-01-26T00:10:00',
  'SAN PABLO AVE / MARIN AVE',
  'T',
  'M',
  None,
  None],
 [2,
  '1644D161-1113-4C4F-BB2E-BF780E7AE73E',
  2,
  1444146408,
  '932858',
  14
```

يمكن أن نلاحظ أن المفتاح 'meta' يحتوي على وصف البيانات والأعمدة. والمفتاح 'data' يحتوي على البيانات. يمكننا استخدام هذه المعلومة لإنشاء ال DataFrame:

```
# تحميل البيانات من JSON
# ثم تسمية الأعمدة
stops = pd.DataFrame(
    stops_dict['data'],
    columns=[c['name'] for c in stops_dict['meta']['view']['columns']])
```

```
    columns=[c['name'] for c in stops_dict['meta']['view']['columns']])
stops
```

d	position	created_at	...	Incident Type	Dispositions	Location - Latitude	Location - Longitude
B912- -4431- C9- :09C32E	1	1444146408	...	T	M	None	None
D161- -4C4F- 2E- :7AE73E	2	1444146408	...	T	M	None	None
ABAB- -488D- 5F- :505872	3	1444146408	...	T	M	None	None
..	...	...	...	...	...	...	...
06ED- -4B0B- 9B- .49F34B	31079	1496269085	...	T	;BM2TWN	None	None
F18D- -441D- 09- DACA7A	31080	1496269085	...	T	;HM4TCS	37.8698757	122.2865508-
D2A4- -4BE7- C6- :9DF838	31081	1496269085	...	1194	;AR	37.86720754	122.2565294-

29208 rows × 15 columns

```
# طباعة اسماء الأعمدة
stops.columns
```

```
Index(['sid', 'id', 'position', 'created_at', 'created_meta', 'updated_at',
       'updated_meta', 'meta', 'Incident Number', 'Call Date/Time', 'Location',
       'Incident Type', 'Dispositions', 'Location - Latitude',
       'Location - Longitude'],
      dtype='object')
```

يحتوي الموقع الذي حصلنا على البيانات منه على المعلومات التالية عن الأعمدة:

العمود	الوصف	النوع
Incident Number	رقم المخالفة، تم توليد هذا الرقم بشكل تلقائي بواسطة برنامج (CAD)	نص
Call Date/Time	تاريخ ووقت المخالفة - الإيقاف	تاريخ + وقت
Location	العنوان العام للمخالفة - الإيقاف	نص
Incident Type	نوع المخالفة يتم توليده أوتوماتيكياً بواسطة (CAD). الحرف (T) يعني إيقاف سيارة من قبل الشرطة. إيقاف سيارة مشبوهة رمز لها بـ (1196). إيقاف مشاة رمز له (1194). إيقاف دراجة بالرمز (1194B)	نص
Dispositions	تصرفات ومعلومات عن الموقوف، مرتبة بالترتيب: أول حرف يعني العرق ، وهو كالتالي: A (آسيوي) B (أسود) H (أسباني) O (آخر) W (أبيض) . الحرف الثاني يعني الجنس: F (أنثى) M (ذكر). الحرف الثالث يعني مدى العمر وهو كالتالي: 1 (أقل من 18) 2 (بين 18-29) 3 (بين 30-39), 4 (أكبر من 40). الحرف الرابع يعني السبب: I (تحقيق) T (إيقاف مروري) R (اشتباه) K (إفراج مشروط) W (مطلوب). الحرف الخامس يعني ما تم تطبيقه على الموقوف: A (اعتقال) C (مخالفه) O (أخرى) W (إنذار). الحرف السادس يوضح إذا تم البحث في السيارة: S (تم البحث) N (لم يتم البحث). معلومات أخرى قد تظهر، منها: P - تقرير الحالة الأولى. M - سرد عن طريق الهاتف فقط. AR - تقرير اعتقال فقط (لم يتم تسليم تقرير القضية). IN - تقرير حادثة. FC - بطاقة ميدانيه. CO - تقرير حادثة تصادم. MH - تقييم نفسي عاجل. TOW - سيارة محجوزة. 0 أو 00000 – الشرطي أوقف لأكثر من خمسة أشخاص	نص
Location - Latitude	خط العرض العام لموقع المكالمة. هذه المعلومة فقط تم البدء بإضافتها في يناير 2017.	رقم
Location - Longitude	خط العرض العام لموقع المكالمة. هذه المعلومة فقط تم البدء بإضافتها في يناير 2017.	رقم

نلاحظ أن الموقع لا يحتوي على وصف لأول ثمانية أعمدة لجدول stops. كون هذه الأعمدة تحتوي على بيانات وصفيه metadata لسنا مهتمين بتحليلها في الوقت الحالي. سنقوم بحذف هذه الأعمدة من الجدول:

```
columns_to_drop = ['sid', 'id', 'position', 'created_at', 'created_meta',
                   'updated_at', 'updated_meta', 'meta']

# هذه الداله تستقبل DataFrame
# وتقوم بحذف الأعمدة غير المرغوبه منها
# تسجل هذه الأعمدة في المتغير الذي سبق تعريفه columns_to_drop
def drop_unneeded_cols(stops):
    return stops.drop(columns=columns_to_drop)

stops = stops.pipe(drop_unneeded_cols)
stops
```

Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
2015-00004825	2015-01-26T00:10:00	SAN PABLO AVE / MARIN AVE	T	M	None	None
2015-00004829	2015-01-26T00:50:00	SAN PABLO AVE / CHANNING WAY	T	M	None	None
2015-00004831	2015-01-26T01:03:00	UNIVERSITY AVE / NINTH ST	T	M	None	None
...	...	...	...	...	...	...
2017-00024245	2017-04-30T22:59:26	UNIVERSITY AVE/6TH ST	T	;BM2TWN	None	None
2017-00024250	2017-04-30T23:19:27	UNIVERSITY AVE / WEST ST	T	;HM4TCS	37.8698757	122.2865508-
2017-00024254	2017-04-30T23:38:34	CHANNING WAY / BOWDITCH ST	1194	;AR	37.86720754	122.2565294-

29208 rows × 7 columns

كما في بيانات المكالمات، سنجيب على الأسئلة التالية:

- **هل توجد بيانات مفقوده؟**
- **هل توجد أي بيانات مفقوده تم تعبئتها؟ (مثلاً كتابة رقم 999 لعمر مجهول أو 12:00 صباحاً لتاريخ مجهول)؟**
- **أي جزء من البيانات أدخلت بواسطة أشخاص حقيقيين؟**

هل توجد بيانات مفقوده ؟

يمكن ان نلاحظ ان هناك الكثير من البيانات المفقودة في عمودي خط الطول والعرض. في وصف البيانات ذكر أن العمودان تم البدء في تعبئتها في يناير 2017:

```
# ستظهر لنا الأسطر التي تحتوي على الأقل على قيمه واحدة مفقوده
null_rows = stops.isnull().any(axis=1)

stops[null_rows]
```

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
0	2015-00004825	2015-01-26T00:10:00	SAN PABLO AVE / MARIN AVE	T	M	None	None
1	2015-00004829	2015-01-26T00:50:00	SAN PABLO AVE / CHANNING WAY	T	M	None	None
2	2015-00004831	2015-01-26T01:03:00	UNIVERSITY AVE / NINTH ST	T	M	None	None

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
...	...	...	...	...	...	...	...
29078	2017-00023764	2017-04-29T01:59:36	M L 2180 KING JR WAY	1194	;BM4IWN	None	None
29180	2017-00024132	2017-04-30T12:54:23	6TH/UNI	1194	;M	None	None
29205	2017-00024245	2017-04-30T22:59:26	UNIVERSITY AVE/6TH ST	T	;BM2TWN	None	None

25067 rows × 7 columns

يمكن أن نتحقق من بقية الأعمدة إذا كانت تحتوي على بيانات مفقوده:

```
# ستظهر لنا الأسطر التي تحتوي على الأقل على قيمه واحدة مفقوده
# دون التحقق من المفقودات في عمودي خطوط الطول والعرض
null_rows = stops.iloc[:, :-2].isnull().any(axis=1)

df_interact(stops[null_rows])
```

بنفس الطريقة السابقة سيظهر لنا جدول تفاعلي.

row 0

	sid	id	position	created_at	created_meta	updated_at	updated_meta	meta	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	L
0	1	29A1B912-A0A9-4431-ADC9-FB375809C32E	1	1444146408	932858	1444146408	932858	None	2015-00004825	2015-01-26T00:10:00	SAN PABLO AVE / MARIN AVE	T	M	
1	2	1644D161-1113-4C4F-BB2E-BF780E7AE73E	2	1444146408	932858	1444146408	932858	None	2015-00004829	2015-01-26T00:50:00	SAN PABLO AVE / CHANNING WAY	T	M	
2	3	5338ABAB-1C96-488D-B55F-6A47AC505872	3	1444146408	932858	1444146408	932858	None	2015-00004831	2015-01-26T01:03:00	UNIVERSITY AVE / NINTH ST	T	M	
3	4	21B6CBE4-9865-460F-97BC-6B26C6EF2FDB	4	1444146408	932858	1444146408	932858	None	2015-00004848	2015-01-26T07:16:00	2000 BLOCK BERKELEY WAY	1194	BM4ICN	
4	5	0D85FA92-80E9-48C2-B409-C3270251CD12	5	1444146408	932858	1444146408	932858	None	2015-00004849	2015-01-26T07:43:00	1700 BLOCK SAN PABLO AVE	1194	BM4ICN	

(63 rows, 7 columns) total

نلاحظ أن أكثر البيانات المفقودة تكون في عمود `Dispositions` "تصرفات ومعلومات عن الموقوف". للأسف، لم يتم التوضيح في شرح البيانات سبب فقدان هذه المعلومات. بما أن عدد البيانات المفقودة 63 مقارنة بمجموع البيانات 25000 في الجدول، سنكمل تنظيف البيانات مع الأخذ بالاعتبار أن البيانات المفقودة قد تؤثر على نتائجنا.

هل توجد أي بيانات مفقوده تم تعبئتها ؟

لا يبدو لدينا أن أي من البيانات المفقودة تم تعبئتها. على عكس بيانات المكالمات، التي تم فيها تقسيم التاريخ والوقت في أعمدة منفصله، في جدول الإيقافات يظهر التاريخ والوقت في عمود واحد.

أي جزء من البيانات أدخلت بواسطة أشخاص حقيقيين ؟

أيضاً، كما في بيانات المكالمات، يبدو أن الكثير من بيانات الإيقافات تم تعبئتها بشكل تلقائي بواسطة الآلة أو تم اختيارها من قوائم محددة مسبقاً بواسطة أشخاص (مثل عمود `Incident Type` "نوع المخالفه").

نلاحظ أن عمود العنوان لا يحتوي على بيانات محددة مسبقاً من قوائم. يمكننا التحقق لنرى بعض الأخطاء الإملائية في كتابة العناوين:

```
stops['Location'].value_counts()
```

```
2200 BLOCK SHATTUCK AVE           229
37.8693028530001~-122.272234021   213
UNIVERSITY AVE / SAN PABLO AVE    202
```

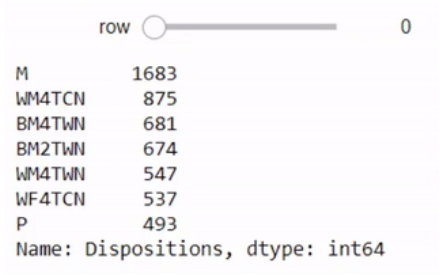
```
                                ...
VALLEY ST / DWIGHT WAY             1
COLLEGE AVE / SIXTY-THIRD ST       1
GRIZZLY PEAK BLVD / MARIN AVE      1
Name: Location, Length: 6393, dtype: int64
```

كما يظهر، يبدو أنه في مرات تم إدخال العنوان كاملاً، أو التقاطعات، ومرات أخرى تم إدخال خطوط الطول والعرض. للأسف ليس لدينا بيانات كاملة لخطوط الطول والعرض لاستخدامها بدلاً من العناوين في هذا العمود. قد نحتاج لتنظيف هذا العمود يدوياً إذا ما أردنا تحليل بياناته.

يمكننا أيضاً التحقق من عمود Dispositions:

```
dispositions = stops['Dispositions'].value_counts()

interact(lambda row=0: dispositions.iloc[row:row+7],
         row=(0, len(dispositions), 7))
```

بنفس الطريقة السابقة سيظهر لنا جدول تفاعلي بالأنواع المختلفة لتصرفات ومعلومات الموقوف



يحتوي العمود على الكثير من التناقضات والاختلافات في تسجيل البيانات. مثلاً، في بعض الأحيان يتم إضافة مسافة قبل إدخال البيانات، وبعض المرات يتم إضافة فاصلة منقوطة في النهاية. بعض الأسطر أيضاً تحتوي على أكثر من إدخال. التنوع في البيانات يوضح أنها قد تكون أدخلت بواسطة أشخاص حقيقيين وليس الآلة، لذا يجب علينا الحذر عند العمل عليها:

```
# امثله على قيم تحتوي على مشاكل
dispositions.iloc[[0, 20, 30, 266, 1027]]
```

```
M           1683
M;           238
 M           176
HF4TWN;       14
 OM4KWS        1
Name: Dispositions, dtype: int64
```

أيضاً، أكثر القيم تكراراً هو M وهو كقيمه أولاً لا تعتبر صحيحه لأنه ليس من قائمة الخيارات في العرق. هذا يقترح أن تم تغير طريقة إدخال البيانات في هذا العمود بعد فترة من بدأ الإدخال، أو أنه مسموح لمدخل البيانات إدخال قيم دون التحقق أن كانت تطابق شروط الإدخال في العمود. على أية حال، سيشكل العمل على هذا العمود تحدي كبير.

يمكننا البدء بخطوات بسيطه لتنظيف هذا العمود عن طريق حذف المسافات الفارغة أن كانت في البداية أو النهاية، حذف الفواصل المنقوطة إذا كانت في آخر الجملة، وإذا تكررت الفاصلة المنقوطة في نفس السطر نستبدها بفاصلة عادية:

```
def clean_dispositions(stops):
    cleaned = (stops['Dispositions']
               .str.strip()             # حذف المسافات الفارغه ان كانت في البدايه او النهايه
               .str.rstrip(';')         # حذف الفواصل المنقوطه اذا كانت في آخر الجمله
               .str.replace(';', ','))  # تبديل باقي الفواصل المنقوطه إلى فواصل
    return stops.assign(Dispositions=cleaned)
```

```
stops_final = (stops
               .pipe(drop_unneeded_cols)
               .pipe(clean_dispositions))
df_interact(stops_final)
```

بنفس الطريقة السابقة سيظهر لنا جدول تفاعلي كالتالي

row 0

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
0	2015-00004825	2015-01-26T00:10:00	SAN PABLO AVE / MARIN AVE	T	M	None	None
1	2015-00004829	2015-01-26T00:50:00	SAN PABLO AVE / CHANNING WAY	T	M	None	None
2	2015-00004831	2015-01-26T01:03:00	UNIVERSITY AVE / NINTH ST	T	M	None	None
3	2015-00004848	2015-01-26T07:16:00	2000 BLOCK BERKELEY WAY	1194	BM4ICN	None	None
4	2015-00004849	2015-01-26T07:43:00	1700 BLOCK SAN PABLO AVE	1194	BM4ICN	None	None

(29208 rows, 7 columns) total

الملخص

كما رأينا، كلا الملفين وضحت كيف أن تنظيف البيانات أحياناً يكون صعباً ومملاً. تنظيف البيانات بشكل كامل يأخذ وقتاً طويل، ولكن عدم تنظيفها قد يوصلنا لنتائج خاطئة. عند مواجهة بيانات يجب أن نحدد خياراتنا ونحقق التوازن في خطوات تنظيف البيانات حتى نحصل على نتائج صحيحه. القرارات التي نتخذها أثناء تنظيف البيانات تأثر على تحليلنا. مثلاً، قررنا عدم تنظيف عمود Location في بيانات الإيقافات، لذا يجب أن نحذر أثناء تحليله. يجب علينا تسجيل كل قرار اتخذناه أثناء تنظيف البيانات ليكون مرجع لنا لاحقاً، والأفضل يكون في ملف جوبتر مع الكود البرمجي كي يكونا كليهما أمامنا وقت المراجعة.

```
stops_final.to_csv('stops.csv', index=False)
```

هيكلة وربط البيانات

الهيكل

هيكل Structure البيانات يعني شكل ملف البيانات. بشكل أبسط، يعني الطريقة التي أدخلت فيها البيانات في الملف. مثلاً، شاهدنا في ملف بيانات المكالمات أن الملف من النوع CSV:

```
!head data/Berkeley_PD_-_Calls_for_Service.csv
```

```
CASENO,OFFENSE,EVENTDT,EVENTTM,CVLEGEND,CVDOW,InDbDate,Block_Location,BLKADDR,City,State
17091420,BURGLARY AUTO,07/23/2017 12:00:00 AM,06:00,BURGLARY - VEHICLE,0,08/29/2017 08:28:05 AM,"2500 LE C
Berkeley, CA
(37.876965, -122.260544)",2500 LE CONTE AVE,Berkeley,CA
17020462,THEFT FROM PERSON,04/13/2017 12:00:00 AM,08:45,LARCENY,4,08/29/2017 08:28:00 AM,"2200 SHATTUCK AV
Berkeley, CA
(37.869363, -122.268028)",2200 SHATTUCK AVE,Berkeley,CA
17050275,BURGLARY AUTO,08/24/2017 12:00:00 AM,18:30,BURGLARY - VEHICLE,4,08/29/2017 08:28:06 AM,"200 UNIVE
Berkeley, CA
(37.865491, -122.310065)",200 UNIVERSITY AVE,Berkeley,CA
```

وبيانات الإيقافات من النوع JSON:

```
# عرض اول واخر خمس اسطر
!head -n 5 data/stops.json
!echo '...'
!tail -n 5 data/stops.json
```

```
{
  "meta" : {
    "view" : {
      "id" : "6e9j-pj9p",
      "name" : "Berkeley PD - Stop Data",
...
, [ 31079, "C2B606ED-7872-4B0B-BC9B-4EF45149F34B", 31079, 1496269085, "932858", 1496269085, "932858", nul
, [ 31080, "8FADF18D-7FE9-441D-8709-7BFEABDACA7A", 31080, 1496269085, "932858", 1496269085, "932858", nul
, [ 31081, "F60BD2A4-8C47-4BE7-B1C6-4934BE9DF838", 31081, 1496269085, "932858", 1496269085, "932858", nul
 ]
}
```

بالطبع هناك أنواع أخرى لأشكال ملف البيانات، هذه قائمة للأكثر استخداماً:

- Comma-Separated Values (CSV) و Tab-Separated Values (TSV). هذان النوعان عادةً ما تحتوي على بيانات بشكل جداول. في هذه الملفات كل سطر يمثل صف; ويفصل بين البيانات بعلامة الفاصلة (,) أو علامة Tab (\t). العمل على هذه الملفات يعتبر سهل جداً لأن شكلها مشابه بشكل كبير لـ DataFrame في بانداز.
- JavaScript Object Format (JSON) ملفات JSON تكون ذات هيكل هرمي وتحتوي على مفاتيح Keys وقيم Values. نحتاج لقراءة الملف بشكل كامل كقاموس Dictionary في لغة بايثون ومن ثم نبحث عن طريقه لاستخراج القيم إلى DataFrame.
- eXtensible Markup Language (XML) و HyperText Markup Language (HTML) هذه الملفات هرمية وتتبع طريقة المفاتيح والقيم. مثال عليها:

```
<?xml version="1.0" encoding="UTF-8"?>
  <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
```

في فصل لاحق، سنستخدم XPath لاستخراج البيانات من هذا النوع من الملفات.

- بيانات الإدخال Log Data. الكثير من البرامج يمكنها تصدير البيانات على شكل نصوص غير مهيكلة، مثلاً:

```
  2005-03-23 23:47:11,663 - sa - INFO - creating an instance of aux_module.Aux
  2005-03-23 23:47:11,665 - sa.aux.Aux - INFO - creating an instance of Aux
  2005-03-23 23:47:11,665 - sa - INFO - created an instance of aux_module.Aux
  2005-03-23 23:47:11,668 - sa - INFO - calling aux_module.Aux.do_something
  2005-03-23 23:47:11,668 - sa.aux.Aux - INFO - doing something
```

في فصل لاحق، سنستخدم التعابير النمطية Regular Expressions لاستخراج البيانات من هذا النوع من الملفات.

الربط

في العادة تكون البيانات مقسمه في أكثر من جدول. مثلاً، الجدول الأول يحتوي على معلومات الأشخاص، والجدول الثاني يحتوي على البريد الإلكتروني لكل شخص:

```
people = pd.DataFrame(
    [["Joey",      "blue",    42,  "M"],
     ["Weiwei",    "blue",    50,  "F"],
     ["Joey",      "green",    8,  "M"],
     ["Karina",    "green",    7,  "F"],
     ["Nhi",       "blue",     3,  "F"],
     ["Sam",       "pink",   -42,  "M"]],
    columns = ["Name", "Color", "Number", "Sex"])

people
```

	Name	Color	Number	Sex
0	Joey	blue	42	M
1	Weiwei	blue	50	F
2	Joey	green	8	M
3	Karina	green	7	F
4	Nhi	blue	3	F
5	Sam	pink	42-	M

```
email = pd.DataFrame(
    [["Deb",  "deborah_nolan@berkeley.edu"],
     ["Sam",  "samlau95@berkeley.edu"],
     ["John", "doe@nope.com"],
     ["Joey", "jegonzal@cs.berkeley.edu"],
     ["Weiwei", "weiwzhang@berkeley.edu"],
     ["Weiwei", "weiwzhang+123@berkeley.edu"],
     ["Karina", "kgoot@berkeley.edu"]],
    columns = ["User Name", "Email"])

email
```

	User Name	Email
0	Deb	deborah_nolan@berkeley.edu
1	Sam	samlau95@berkeley.edu

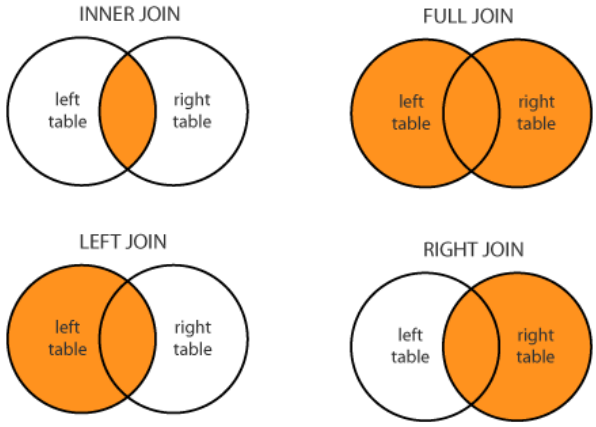
	User Name	Email
2	John	doe@nope.com
3	Joey	jegonzal@cs.berkeley.edu
4	Weiwei	weiwzhang@berkeley.edu
5	Weiwei	weiwzhang+123@berkeley.edu
6	Karina	kgoot@berkeley.edu

لربط كل شخص ببريده الإلكتروني، يمكننا جمع الجدولين في جدول واحد والربط بينهم بواسطة عمود اسم المستخدم. يجب أن نقرر ما نفعله للأشخاص الذين يظهرون في جدول دون الآخر. مثلاً، Fernando ظهر في جدول الأشخاص people ولم يظهر في جدول البريد الإلكتروني email. لدينا أنواع مختلفة من خيارات الربط Join، ولكل نوع طريقه مختلفة في التعامل مع البيانات المفقودة في جدول عن الآخر. أحد أشهر أنواع الربط هي *Inner Join الربط الداخلي*، وفيه أي سطر لا يوجد له قيمه في الجدول الآخر يتم حذفه:

```
# Fernando, Nhi, Deb, و John لا يظهران في جدول البريد الإلكتروني لذا تم حذفهم
people.merge(email, how='inner', left_on='Name', right_on='User Name')
```

	Namr	Color	Number	Sex	User Name	Email
0	Joey	blue	42	M	Joey	jegonzal@cs.berkeley.edu
1	Joey	green	8	M	Joey	jegonzal@cs.berkeley.edu
2	Weiwei	blue	50	F	Weiwei	weiwzhang@berkeley.edu
3	Weiwei	blue	50	F	Weiwei	weiwzhang+123@berkeley.edu
4	Karina	green	7	F	Karina	kgoot@berkeley.edu
5	Sam	pink	42-	M	Sam	samlau95@berkeley.edu

توجد أربع أنواع للربط التي تستخدم بشكل معتاد ودائم: الربط الداخلي Inner Join، الربط الكامل Full Join (و أحياناً يطلق عليه الخارجي Outer Join)، الربط اليساري Left Join, الربط اليميني Right Join. في الأسفل رسم بياني يوضح الفرق بين كل نوع.



شرح مبسط للفرق بين كل نوع. أولاً، حددنا الجدول على اليسار هو people وعلى اليمين هو emails. لدينا شرط في الأمر السابق، وهو أن يتوافق كل اسم left_on='Name' في جدول الأشخاص people مع اسم المستخدم right_on='User Name' في جدول البريد الإلكتروني email. الآن نقوم بالربط ولك يجب علينا تحديد النوع في المتغير how في المثال اخترنا inner:

```
people.merge(email, how='inner', left_on='Name', right_on='User Name')
```

- Inner Join الربط الداخلي: سيتحقق الشرط من توفر القيم في كلا الجدولين. في حال وجد قيمه موجودة في جدول دون الآخر فسيتم تجاهلها.
- Full Join (Outer) الربط الكامل (الخارجي): يربط بين كل القيم سواء طابقت للشرط الذي حددناه أو لم تطابق، القيم التي تتوفر في جدول دون الآخر يتم إضافتها أيضاً وتضاف قيمه NaN للأعمدة التي لا تحتوي على قيم.
- Left Join الربط اليساري: يتم استخدام كل القيم في الجدول على اليسار (في مثالنا هنا جدول people) سواء تحتوي على قيم في الجدول الآخر أو لا، بينما يتم جلب القيم التي توافق الشرط الذي حددنا من الجدول على اليمين (في مثالنا هنا جدول email).
- Right Join الربط اليميني: عكس الربط اليساري، في مثالنا هنا يتم جلب جميع القيم في الجدول email وفي الجدول اليساري people يتم استخدام فقط القيم التي توافق الشرط، وهي القيم التي تحتوي على أسماء مستخدمين موجودة في الجدول email.

قائمة المراجعة لهيكل البيانات

بعد مراجعتك لهيكل البيانات، يجب عليك الإجابة على الأسئلة التالية. سنجيب عن الأسئلة باستخدام بيانات المكالمات والإيقافات.

هل البيانات منسقه بشكل أو ترميز أساسي/عام؟

الأنواع الأساسيه تشمل:

- البيانات المجدولة: CSV, TSV, Excel, SQL.
- البيانات الهرمية: JSON, XML.

بيانات المكالمات جاءت على شكل CSV وبيانات الإيقافات على شكل JSON.

هل البيانات مُرتبه على شكل صفوف؟ إذا كان الجواب لا، هل يمكننا معرفة الأسطر عند مراجعة البيانات؟

بيانات المكالمات جاءت على شكل أسطر، بينما قمنا باستخراج الأسطر في بيانات الإيقافات.

هل البيانات هرمية؟ هل يمكننا تفكيكها؟

بيانات المكالمات ليست هرمية، لم يتعين علينا العمل بشكل كبير على بيانات الإيقافات لتفكيك هرميتها.

هل أضيف للبيانات أي مراجع؟ إذا كان كذلك، هل يمكننا ربطها بالبيانات؟

بيانات المكالمات أضافت جدول أيام الأسبوع كمرجع. ربط وجمع الجدولين معاً أضاف لنا أيام الأسبوع لكل مخالفه. لم يتم ذكر أي مرجع لبيانات الإيقافات.

ما هي الأعمدة في كل جدول؟ ما نوع كل عمود؟

تم وصف وشرح كل نوع من الأعمدة في خطوة تنظيف البيانات لكل جدول. هنا لبيانات المكالمات، وهنا لبيانات الإيقافات.

جودة البيانات

جودة البيانات تعني ما يمثله كل سطر في بياناتك. مثلاً، في بيانات المكالمات كل سطر يمثل حاله واحدة لمكالمة الشرطة:

```
calls = pd.read_csv('data/calls.csv')
calls.head()
```

	CASENO	OFFENSE	CVLEGEND	BLKADDR	EVENTDTTM	Latitude	Longitude	Day
0	17091420	BURGLARY AUTO	BURGLARY - VEHICLE	LE 2500 CONTE AVE	23/07/2017 6:00	37.876965	122.260544-	Sunday
1	17038302	BURGLARY AUTO	BURGLARY - VEHICLE	BOWDITCH STREET & CHANNING WAY	02/07/2017 22:00	37.867209	122.256554-	Sunday
2	17049346	THEFT MISD. ((UNDER $950	LARCENY	2900 CHANNING WAY	20/08/2017 23:20	37.867948	122.250664-	Sunday
3	17091319	THEFT MISD. ((UNDER $950	LARCENY	2100 RUSSELL ST	09/07/2017 4:15	37.856719	122.266672-	Sunday
4	17044238	DISTURBANCE	DISORDERLY CONDUCT	TELEGRAPH AVENUE & DURANT AVE	30/07/2017 1:16	37.867816	122.258994-	Sunday

في بيانات الإيقافات كل سطر يعني حالة إيقاف من قبل الشرطة:

```
stops = pd.read_csv('data/stops.csv', parse_dates=[1], infer_datetime_format=True)
stops.head()
```

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
0	2015-00004825	26/01/2015 0:10	SAN PABLO AVE / MARIN AVE	T	M	NaN	NaN
1	2015-00004829	26/01/2015 0:50	SAN PABLO AVE / CHANNING WAY	T	M	NaN	NaN
2	2015-00004831	26/01/2015 1:03	UNIVERSITY AVE / NINTH ST	T	M	NaN	NaN
3	2015-00004848	26/01/2015 7:16	BLOCK 2000 BERKELEY WAY	1194	BM4ICN	NaN	NaN

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
4	2015-00004849	26/01/2015 7:43	BLOCK SAN 1700 PABLO AVE	1194	BM4ICN	NaN	NaN

يمكن أن تأتينا بيانات الإيقافات بشكل آخر كالتالي:

```
(stops
 .groupby(stops['Call Date/Time'].dt.date)
 .size()
 .rename('Num Incidents')
 .to_frame()
)
```

Call Date/Time	Num Incidents
26/01/2015	46
27/01/2015	57
28/01/2015	56
...	...
28/04/2017	82
29/04/2017	86
30/04/2017	59

825 rows × 1 columns

في هذه الحالة، كل سطر يعني حالات الإيقاف التي تمت في ذلك التاريخ. يمكن أن نوصف هذا الجدول بجدول ذو جودة رديئه مقارنة بالجدول الذي سبقه. من المهم معرفة جودة البيانات لأنها تحدد نوع التحليل الذي سوف تقوم به. بشكل عام، كل ما كانت جودة البيانات أفضل كان العمل عليها أفضل، على الرغم أن بإمكاننا استخدام التجميع والجداول المحورية لتحويل بيانات من جيدة إلى رديئه، يمكننا تحويل بيانات رديئه إلى جيدة باستخدام بعض الأدوات.

قائمة المراجعة لجودة البيانات

يجب عليك الإجابة على الأسئلة التالية بعد التحقق من جودة البيانات. سنجيب على الأسئلة لبيانات المكالمات والإيقافات.

ماذا يمثل كل سطر؟

في بيانات المكالمات، كل سطر يمثل حاله واحدة لمكالمة الشرطة. في بيانات الإيقافات، كل سطر يمثل حالة إيقاف من قبل الشرطة.

هل كل البيانات تملك نفس الجودة؟ (أحياناً الجدول يحتوي على أسطر مجاميع)

نعم، لجدولي المكالمات والإيقافات.

إذا كانت البيانات عبارة عن مجاميع، كيف تم جمعها؟ العينات والمتوسطات أحد أكثر أمثلة التجميع استخداماً

على حسب رؤيتنا للبيانات، لا تحتوي على مجاميع. سنأخذ بعين الاعتبار أن بيانات الموقع هي عبارة عن أسماء الأحياء بدلاً من عناوين معينة.

أي نوع من التجميع يمكننا تطبيقه على البيانات؟

مثلاً، أحد المعلومات التي يمكننا الاستفادة منها هي التجميع بين الأشخاص ومواقع البيانات أو الأحداث لإيجاد المجاميع خلال فترة معينة.

في هذا المثال، يمكننا الجمع بناءاً على التاريخ أو الوقت. مثلا، يمكننا إيجاد الأوقات التي يكثر فيها الحالات يومياً. يمكننا أيضاً الجمع بواسطة نوع المخالفات وموقعها لمعرفة في أي مواقع بيركلي تكثر الحوادث.

مدى البيانات

تحتاج لتحميل البيانات بعد القيام بعملية التنظيف في درسي تنظيف بيانات المكالمات و تنظيف بيانات الإيقافات، أو تحميلها هنا:

- بيانات المكالمات: هنا.
- بيانات الإيقافات: هنا.

مدى البيانات هو مدى قدرة البيانات على تغطية ما نحتاجه للقيام بالتحليل. نحاول الإجابة على الأسئلة التالية لمعرفة مدى بياناتنا:

يقصد الكاتب بالمدى Scope ما إذا كانت البيانات شاملة ومفيده لنقوم بتحليلها ونجيب على الأسئلة التي وضعناها باستخدام البيانات.

هل تغطي البيانات المواضيع التي تهمنا؟

مثلاً، بيانات المكالمات والإيقافات تحتوي على جميع الحالات في مدينة بيركلي. إذا كنا مهتمين بحالات الجرائم في ولاية كاليفورنيا، فهذه البيانات لن تفيدنا كونها محدودة المدى.

بشكل عام، كلما كان المدى أكبر، كلما أصبح أكثر فائدة لأن بإمكاننا فلترة البيانات الكبيرة لعده مداءات صغيرة ولا يمكننا عمل العكس، تحويل المداءات الصغيرة إلى كبيرة. مثلاً، إذا كانت لدينا بيانات الإيقافات في الولايات المتحدة، يمكننا فلترة وتقسيم البيانات لنقوم بتحليلنا على بيانات بيركلي.

خذ بعين الاعتبار، أن المدى مصطلح عام وليس مخصص فقط للمواقع الجغرافية. مثلاً، يمكن استخدامه مع الوقت، بيانات المكالمات تحتوي على بيانات 180 يوم فقط.

عاده ما نقوم بالتحقق من المدى عندما نقوم بالكشف عن طريقة توليد البيانات ونقوم بالتأكد من المدى في خطوة التحليل الاستكشافي للبيانات.
لنتحقق من مدى الموقع الجغرافي والوقت في بيانات المكالمات:

```
calls = pd.read_csv('data/calls.csv', parse_dates=['EVENTDTTM'], infer_datetime_format=True)
stops = pd.read_csv('data/stops.csv', parse_dates=[1], infer_datetime_format=True)

calls
```

	CASENO	OFFENSE	CVLEGEND	BLKADDR	EVENTDTTM	Latitude	Longitude	Day
0	17091420	BURGLARY AUTO	BURGLARY - VEHICLE	LE 2500 CONTE AVE	23/07/2017 6:00	37.876965	122.260544-	Sunday
1	17038302	BURGLARY AUTO	BURGLARY - VEHICLE	BOWDITCH STREET & CHANNING WAY	02/07/2017 22:00	37.867209	122.256554-	Sunday
2	17049346	THEFT MISD. ((UNDER $950	LARCENY	2900 CHANNING WAY	20/08/2017 23:20	37.867948	122.250664-	Sunday
...	...	...	...	...	...	...	...	...
5505	17021604	IDENTITY THEFT	FRAUD	100 MONTROSE RD	31/03/2017 0:00	37.896218	122.270671-	Friday
5506	17033201	DISTURBANCE	DISORDERLY CONDUCT	2300 COLLEGE AVE	09/06/2017 22:34	37.868957	122.254552-	Friday
5507	17047247	BURGLARY AUTO	BURGLARY - VEHICLE	UNIVERSITY AVENUE & CHESTNUT ST	11/08/2017 20:00	37.869679	122.288038-	Friday

5508 rows × 8 columns

```
# عرض اقرب وابعد تاريخ في بيانات المكالمات
calls['EVENTDTTM'].dt.date.sort_values()
```

```
1384    2017-03-02
1264    2017-03-02
1408    2017-03-02
           ...
3516    2017-08-28
3409    2017-08-28
3631    2017-08-28
Name: EVENTDTTM, Length: 5508, dtype: object
```

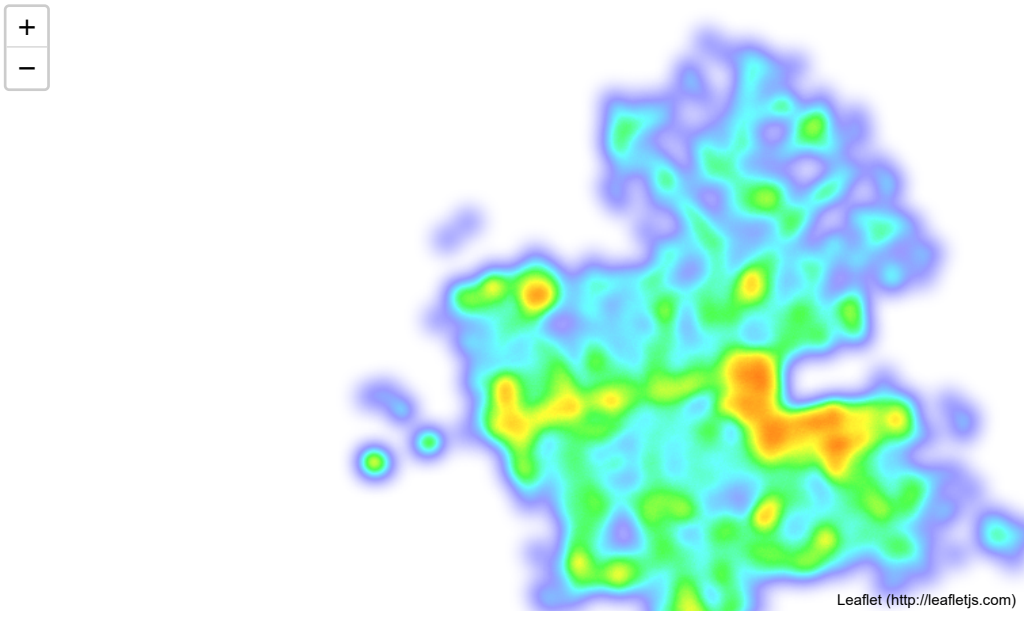
```
calls['EVENTDTTM'].dt.date.max() - calls['EVENTDTTM'].dt.date.min()
```

```
datetime.timedelta(179)
```

يحتوي الجدول على بيانات 179 يوم وهي قريباً بشكل كبير إلى 180 يوماً التي ذكرت في وصف البيانات، لذا نتوقع أن أحد الأيام مر دون القيام بأي مكالمه، لذا نتوقع أن اليوم الذي لم يحصل فيه أي مكالمه اما 14 أبريل 2017 او 29 أغسطس 2017

للتحقق من مدى المواقع الجغرافيه سنستخدم الخريطه:

```
import folium # سنستخدم مكتبة Folium لرسم الخريطه
import folium.plugins


SF_COORDINATES = (37.87, -122.28)
sf_map = folium.Map(location=SF_COORDINATES, zoom_start=13)
locs = calls[['Latitude', 'Longitude']].astype('float').dropna().to_numpy()
heatmap = folium.plugins.HeatMap(locs.tolist(), radius = 10)
sf_map.add_child(heatmap)
```



مع القليل من الإستثناءات، بيانات المكالمات تغطي مدينة بيركلي بشكل كامل. نلاحظ ان اكثر المكالمات كانت من وسط مدينة بيركلي وجنوب الحرم الجامعي لجامعة بيركلي.

لنتحقق من بيانات الإيقافات لتحديد مداها الزمني والجغرافي:

```
stops
```

	Incident Number	Call Date/Time	Location	Incident Type	Dispositions	Location - Latitude	Location - Longitude
0	2015-00004825	26/01/2015 0:10	SAN PABLO AVE / MARIN AVE	T	M	NaN	NaN
1	2015-00004829	26/01/2015 0:50	SAN PABLO AVE / CHANNING WAY	T	M	NaN	NaN
2	2015-00004831	26/01/2015 1:03	UNIVERSITY AVE / NINTH ST	T	M	NaN	NaN
...	...	...	...	...	...	...	...
29205	2017-00024245	30/04/2017 22:59	UNIVERSITY AVE/6TH ST	T	BM2TWN	NaN	NaN
29206	2017-00024250	30/04/2017 23:19	UNIVERSITY AVE / WEST ST	T	HM4TCS	37.869876	122.286551-
29207	2017-00024254	30/04/2017 23:38	CHANNING WAY / BOWDITCH ST	1194	AR	37.867208	122.256529-

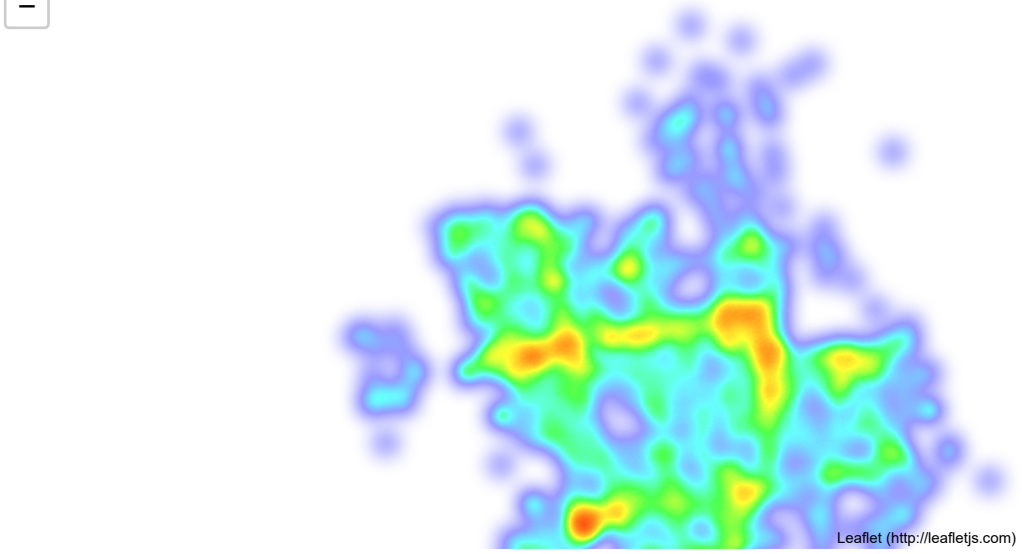
29208 rows × 7 columns

```
stops['Call Date/Time'].dt.date.sort_values()
```

```
0        2015-01-26
25       2015-01-26
26       2015-01-26
            ...
29175    2017-04-30
29177    2017-04-30
29207    2017-04-30
Name: Call Date/Time, Length: 29208, dtype: object
```

كما ذُكر لنا، تم البدء بتجميع البيانات من تاريخ 26 يناير 2015. ويبدو ان تم تحميل البيانات في بداية شهر مايو 2017 كون ان اخر تاريخ في البيانات كان 30 ابريل 2017. لنقم برسم الخريطة للتحقق من المدى الجغرافي:

```
SF_COORDINATES = (37.87, -122.28)
sf_map = folium.Map(location=SF_COORDINATES, zoom_start=13)
locs = stops[['Location - Latitude', 'Location - Longitude']].astype('float').dropna().to_numpy()
heatmap = folium.plugins.HeatMap(locs.tolist(), radius = 10)
sf_map.add_child(heatmap)
```



تأكد لنا الخريطة أن بيانات الإيقافات هي لمدينة بيركلي، وان أكثر الإيقافات كانت في وسط وغرب المدينة.

زمانية البيانات

يقصد بزمانية البيانات هنا بكيفية بناء الوقت والتاريخ في البيانات. نحاول الإجابة عن الأسئلة التالية:

ما معنى أعمدة التاريخ والوقت في البيانات؟

في بيانات المكالمات والإيقافات، بيانات التاريخ والوقت توضح متى تم إجراء المكالمة أو متى تم الإيقاف بواسطة الشرطة. ولكن، بيانات الإيقافات تحتوي أيضاً على أعمدة تاريخ ووقت توضح متى تم إدخال القضية إلى قاعدة البيانات والتي قمنا بحذفها أثناء قيمنا بتنظيف البيانات لأننا رأينا عدم أهميتها لتحليلنا.

أيضاً، يجب علينا الحذر والتحقق من المنطقة الزمنية والتوقيت الصيفي لأعمدة الوقت والتاريخ عند العمل على بيانات تم سحبها من أكثر من موقع.

كيف تم تمثيل التاريخ والوقت في البيانات؟

رغم أن الولايات المتحدة تستخدم النمط MM/DD/YYYY، الكثير من دول العالم تستخدم DD/MM/YYYY. هناك الكثير من الأنماط الأخرى حول العالم ويجب الحذر ومعرفة تلك الأنماط أثناء تحليل البيانات.

في بيانات المكالمات والإيقافات، التواريخ تأتي بالنمط MM/DD/YYYY.

هل هناك انماط زمنيه قد تظهر كقيم غير موجوده؟

بعض البرامج تستخدم أنماط سبق تعرفيها كقيم غير موجودة Null. مثلاً برنامج أكسل يستخدم التاريخ Jan 1st, 1990 ، ونفس البرنامج على أجهزة الماكنتوش يستخدم التاريخ Jan 1st, 1904. الكثير من البرامج الأخرى تقوم بإنشاء وقت وتاريخ بشكل تلقائي 12:00am Jan 1st, 1970 أو 11:59pm Dec 31st, 1969 كون ذلك هو نمط البداية بتوقيت يونكس. إذا لاحظت تكرار هذه القيمة أكثر من مرة في بياناتك، فيجب عليك الحذر ومراجعة مصدر البيانات. لا تحتوي بيانات المكالمات والإيقافات على أي من هذه القيم المشبوهة.

مدى ثقتنا بالبيانات

نطلق على بيانات أنها موثوقة إذا كنا نعتقد أنها تصف الحقيقة بشكل كامل. عادةً البيانات غير الموثوقة تحتوي على التالي:

أرقام غير واقعية أو غير حقيقية

مثلاً، احتواء البيانات على تواريخ مستقبليه، مواقع جغرافية غير موجودة، أرقام سلبية أو قيم شاذة كثيره.

قيم تعتمد على قيم أخرى لكن لا تتوافق معها

مثلاً، تاريخ الميلاد والعمر لا يتطابقان.

بيانات مُدخله يدوياً

كما رأينا سابقاً، هذا النوع من البيانات عاده ما يحتوي على الكثير من الأخطاء الإملائية والتناقضات.

علامات واضحة بوجود بيانات مزورة

مثلاً، تكرار بالأسماء، أو معلومات بريد إلكتروني تبدو غير حقيقة.

لاحظ التشابه الكثير بعمليات تنظيف البيانات. كما ذكرنا سابقاً، عادةً ما نتنقل بين تنظيف البيانات والتحليل الاستكشافي للبيانات، خاصة عندما نحدد مدى ثقتنا بالبيانات. مثلاً، الرسم البياني عادةً ما يساعد باكتشاف القيم الغريبة في البيانات:

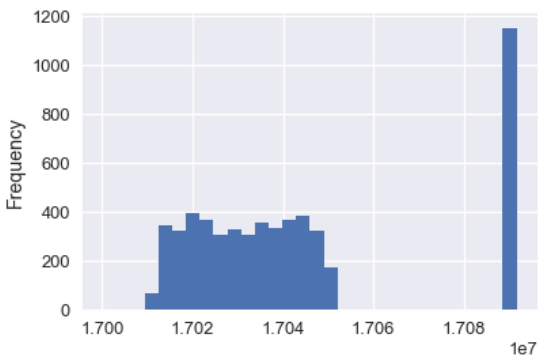
```
calls = pd.read_csv('data/calls.csv')
calls.head()
```

	CASENO	OFFENSE	EVENTDT	EVENTTM	...	BLKADDR	Latitude	Longitude	Day
0	17091420	BURGLARY AUTO	07/23/2017 12:00:00 AM	6:00	...	LE 2500 CONTE AVE	37.876965	122.260544-	Sunday
1	17038302	BURGLARY AUTO	07/02/2017 0:00	22:00	...	BOWDITCH STREET & CHANNING WAY	37.867209	122.256554-	Sunday
2	17049346	THEFT MISD. ((UNDER $950	08/20/2017 12:00:00 AM	23:20	...	2900 CHANNING WAY	37.867948	122.250664-	Sunday
3	17091319	THEFT MISD. ((UNDER $950	07/09/2017 0:00	4:15	...	2100 RUSSELL ST	37.856719	122.266672-	Sunday
4	17044238	DISTURBANCE	07/30/2017 12:00:00 AM	1:16	...	TELEGRAPH AVENUE & DURANT AVE	37.867816	122.258994-	Sunday

5 rows × 9 columns

```
calls['CASENO'].plot.hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1ebb2898>
```



لاحظ المجموعات الغير متوقعة بين 17030000 و 1709000. عن طريق رسم توزيع البيانات لأرقام الحالات، نلاحظ مباشرةً الشذوذ في البيانات. في هذه الحالة، يمكننا توقع أنه يوجد فريقين مختلفين في قسم الشرطة يستخدمون أنواع مختلفة لأرقام الحالات في مكالماتهم.

استكشاف البيانات عادةً ما يظهر أسباب الشذوذ فيها، إذا كان بالإمكان إصلاحها، يمكننا تطبيق تقنيات تنظيف البيانات عليها.

تصوير البيانات

مقدمة

> هناك سحر في الرسوم البيانية. المنحنى في الرسم يظهر بشكل سريع الحالة — قصة حياة وباء، الهلع، أو مرحلة الرخاء. المنحنى ينبه العقل، يثير الخيال، يقنعه.
>
> — هنري دي هوبارد

التصوير والرسوم البيانية أداه أساسية في كل خطوه من خطوات التحليل في علم البيانات، من التنظيف حتى التحليل الاستكشافي للبيانات إلى الوصول لنتائج وتوقعات. لأن العقل البشري متطور للغاية للإدراك البصري، اختيار الرسم البياني الصحيح يظهر لنا الاتجاهات والشذوذ في البيانات بشكل أفضل من الوصف النصي.

لتستخدم تصوير البيانات بشكل فعال، يجب أن تكون خبيراً في أدوات البرمجة التي تنشئ الرسوم البيانية ومبادئ تصوير ورسم البيانات. في هذا الفصل سنتعرف على seaborn و matplotlib، الأدوات التي اخترناها للرسم. سنتعلم أيضا كيف نكتشف الرسوم البيانية المضللة وكيف نحسن ممن الرسوم البيانية باستخدام تحويل، تبسيط، وتقليل أبعاد البيانات.

البيانات الكمية

تصوير البيانات الكمية

عادة ما نستخدم أنواع مختلفة من الرسوم البيانية لتصوير البيانات الكمية (الرقمية) والنوعية (الاسمية).

في البيانات الكمية، عادة ما نستخدم المدرج التكراري Histogram، مخطط الصندوق Box Plot، ومخطط التشتت Scatter Plot.

يمكننا استخدام مكتبة Seaborn للرسوم البيانية لرسم تلك الرسوم في بايثون. سنستخدم بيانات تحتوي على معلومات رُكاب سفينة تايتانيك.

```
# جلب المكتبه
import seaborn as sns

# تطبيق الإعدادات الإفتراضيه لثيم Seaborn
sns.set()

# تحميل وتعريف البيانات وحذف الحقول الفارغه
ti = sns.load_dataset('titanic').dropna().reset_index(drop=True)


df_interact(ti)
```

row 65

col 6

	fare	embarked	class	who	adult_male	deck	embark_town	alive
65	28.50	S	First	man	True	C	Southampton	no
66	153.46	S	First	man	True	C	Southampton	no
67	66.60	S	First	man	True	C	Southampton	no
68	134.50	C	First	woman	False	E	Cherbourg	yes
69	26.00	S	Second	child	False	F	Southampton	yes

```
(182 rows, 15 columns) total
```

كما في الفصل السابق، عرف الكاتب دالة تقوم بإظهار أجزاء من البيانات مع إمكانية التنقل بينها بواسطة الشريطين في الأعلى

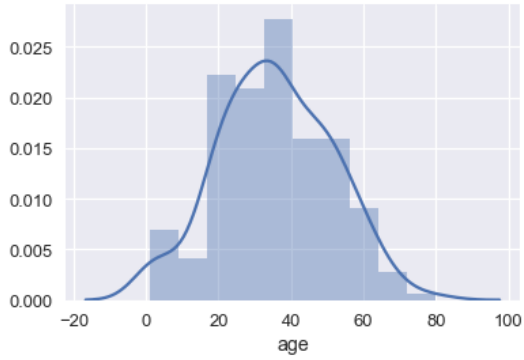
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual

def df_interact(df):
    def peek(row=0, col=0):
```

```
    def peek(row=0, col=0):
        return df.iloc[row:row + 5, col:col + 8]
    interact(peek, row=(0, len(df), 5), col=(0, len(df.columns) - 6))
    print('({} rows, {} columns) total'.format(df.shape[0], df.shape[1]))
```

المدرج التكراري

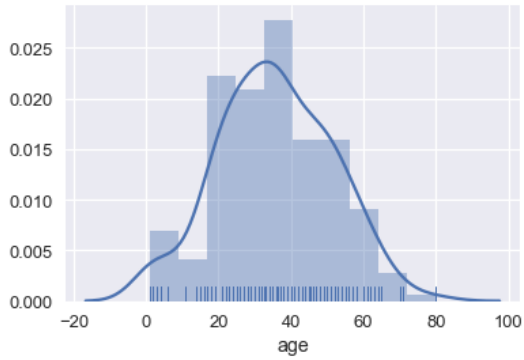
نرى أن كل سطر في البيانات هو معلومات لراكب. كل سطر يحتوي على عمر الراكب والقيمة التي دفعها على التذكرة. لنرسم عمود العمر باستخدام المدرج التكراري. يمكننا استخدام دالة `distplot` الموجودة في Seaborn:

```
# اضافة ; في النهاية تخبر جوبتر أن لا يظهر النتيجة التاليه
# <matplotlib.axes._subplots.AxesSubplot>
# بل يظهر لنا الرسم البياني مباشره
sns.distplot(ti['age']);
```



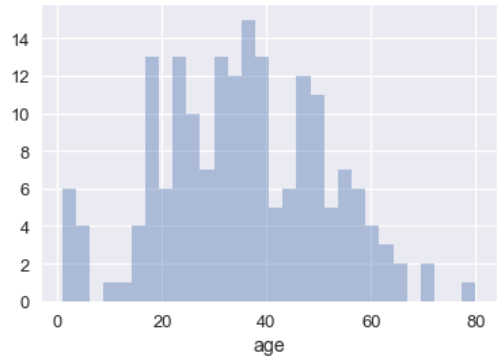
بشكل تلقائي، دالة `distplot` في Seaborn ستُظهر منحنى يناسب توزيع البيانات لدينا. يمكننا إظهار `Rug Plot` والتي تقوم بتحديد أماكن توزيع البيانات على المحور x :

```
sns.distplot(ti['age'], rug=True);
```



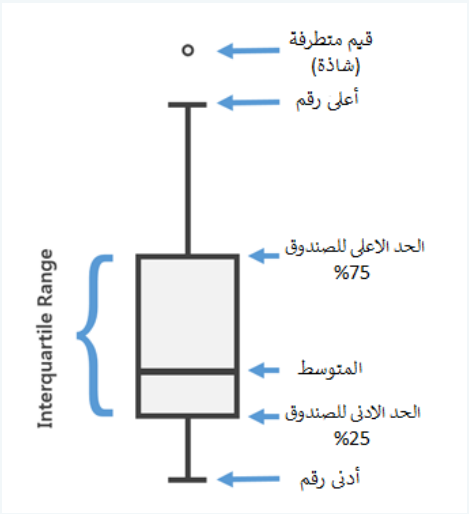
يمكننا أيضا التحكم بالتوزيع. بالتعديل على عدد البيانات لكل مجموعة:

```
sns.distplot(ti['age'], kde=False, bins=30);
```



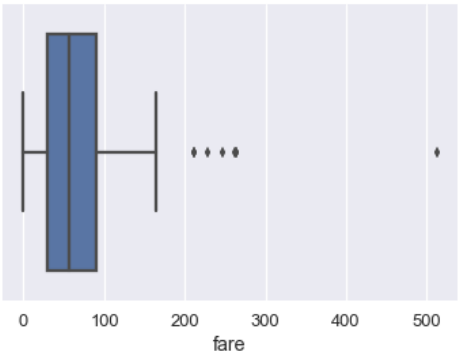
مخطط الصندوق

يعتبر مخطط الصندوق طريقة مناسبة لمعرفة أين تجتمع البيانات. عادةً، نستخدم النسبة 25 و 75 كنقطتي بداية ونهاية للصندوق ونرسم خطاً عند النسبة 50 (المتوسط)، نرسم خارجها خطين لبقية البيانات دون القيم المتطرفة والتي يرمز لها بنقاط خارج الخطين.



يمكن تبسيط مخطط الصندوق بالرسمه التاليه:
مصدر الصورة

```
sns.boxplot(x='fare', data=ti);
```



عادةً ما يستخدم الانحراف الربيعي IQR (Inter-Quartile Range) لتحديد أي النقاط يمكن حسابها كنقاط شاذة في مخطط الصندوق. IQR هو الفرق بين بيانات النسبة 75 و 25:

```
lower, upper = np.percentile(ti['fare'], [25, 75])
iqr = upper - lower
iqr
```

```
60.299999999999997
```

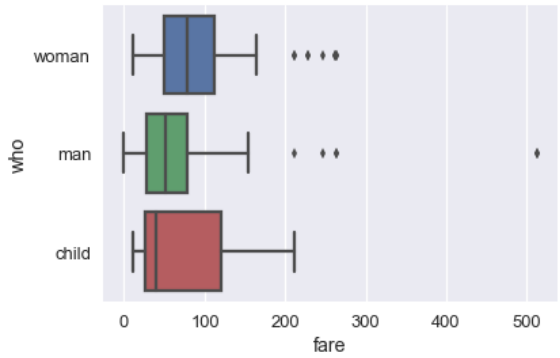
القيم اكبر من $1.5 \times \text{IQR}$ تكون أعلى من نسبة 75% والقيم أقل من $1.5 \times \text{IQR}$ تكون أسفل النسبة 25% ويتم اعتبارها قيم شاذة ويمكننا مشاهدتها في الرسم البياني لمخطط الصندوق السابق:

```
upper_cutoff = upper + 1.5 * iqr
lower_cutoff = lower - 1.5 * iqr
upper_cutoff, lower_cutoff
```

```
(180.44999999999999, -60.749999999999986)
```

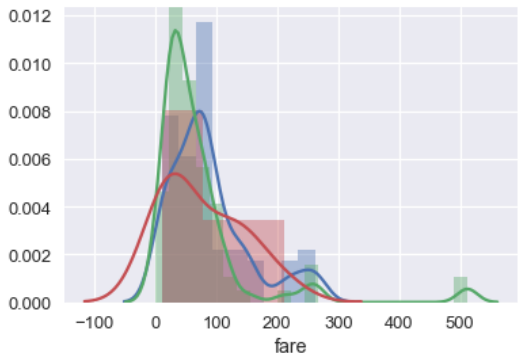
بالرغم أن المدرج التكراري يظهر كل البيانات مرة واحدة، يسهل علينا فهم مخطط الصندوق عندما نقسمه حسب نوع البيانات. مثلاً، يمكننا رسم مخطط الصندوق لكل نوع من الزكاب:

```
sns.boxplot(x='fare', y='who', data=ti);
```



يسهل علينا فهم مخطط الصندوق عند تقسيمه على عكس المدرج التكراري، التالي رسم لنفس البيانات وتقسيمها ولكن في مدرج تكراري:

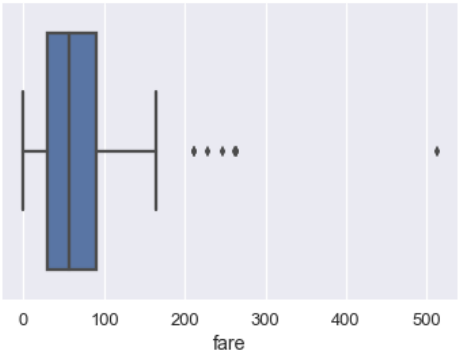
```
sns.distplot(ti.loc[ti['who'] == 'woman', 'fare'])
sns.distplot(ti.loc[ti['who'] == 'man', 'fare'])
sns.distplot(ti.loc[ti['who'] == 'child', 'fare']);
```



نقطة بسيطه عن استخدام مكتبة Seaborn

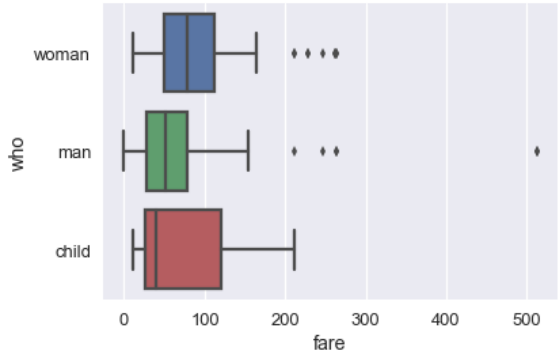
ربما لاحظت أن الدالة `boxplot` أنشأت مخطط صندوق منفصل لكل نوع داخل العمود `who` وكانت أسهل للكتابة على عكس طريقة الرسم في المدرج التكراري. على الرغم مان `sns.distplot` تستقبل البيانات كمصفوفات، الكثير من الدوال في `Seaborn` يمكنها قبول البيانات التي على شكل `DataFrame` ونحدد أي الأعمدة على المحورين `x` و `y`. مثلاً:

```
# ارسم العمود fare في بيانات ti على المحور x
sns.boxplot(x='fare', data=ti);
```



عندما تكون نوع البيانات في العمود نوعية (العمود 'who' يحتوي على 'men', 'women' و 'child')، مكتبة Seaborn ستقوم أوتوماتيكيا بتقسيم البيانات حسب نوعها قبل رسمها. لذا لا نحتاج لفلترة البيانات حسب نوعها كما فعلنا في الدالة sns.distplot:

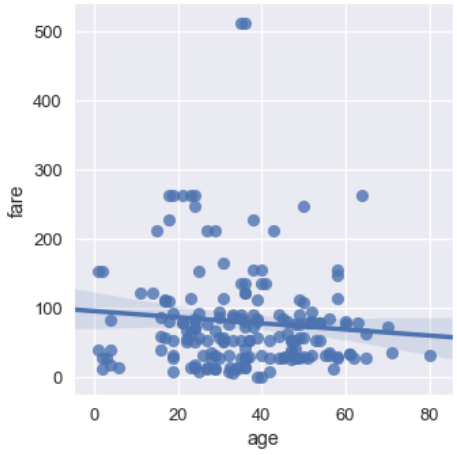
```
# fare (رقمي) على المحور x
# who (نوعي) على المحور y
sns.boxplot(x='fare', y='who', data=ti);
```



مخطط التشتت

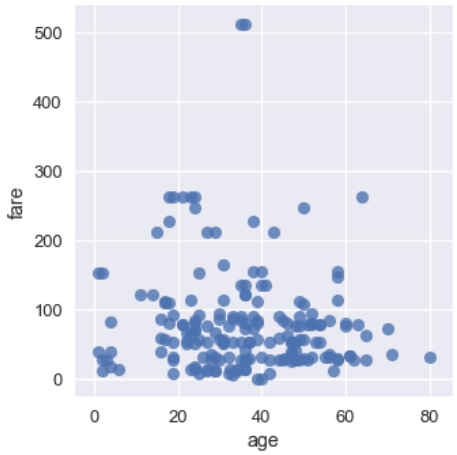
مخطط التشتت يستخدم للمقارنة بين قيمتين كمية (رقمية). يمكننا المقارنة بين العامودان age و fare في بيانات تايتانيك:

```
sns.lmplot(x='age', y='fare', data=ti);
```



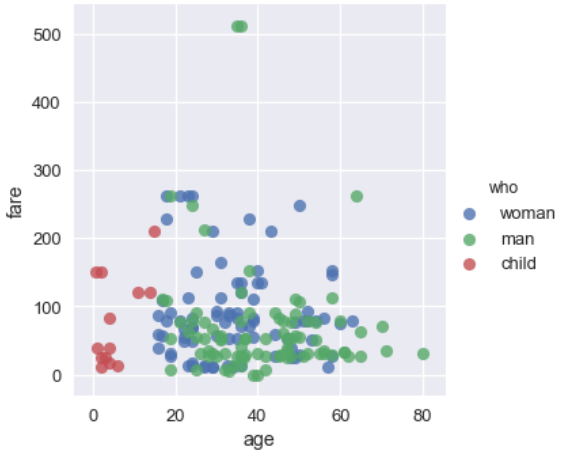
بشكل تلقائي تقوم Seaborn بإضافة خط الانحدار للرسم البياني ومجال ثقة حول خط الانحدار والذي يظهر بلون ازرق شفاف. في حالتنا هذه لا يبدو أن خط الانحدار صحيح لذا سنقوم بإخفائه:

```
sns.lmplot(x='age', y='fare', data=ti, fit_reg=False);
```



يمكننا تلوين النقاط حسب نوعها. لنستخدم العمود who مرة أخرى:

```
sns.lmplot(x='age', y='fare', hue='who', data=ti, fit_reg=False);
```



من الرسم البياني السابق يمكننا ملاحظة أن أكثر الرُكاب دون ال18 سنة تم تحديدهم كأطفال child. لا يبدو أن هناك فرق في سعر التذكرة fare بين الذكور والإناث، على الرغم أن أعلى تذكرتين سعراً تم شرائها من قبل ذكران.

البيانات النوعية

تصوير البيانات النوعية

للبيانات النوعية أو التصنيفية، عادة ما نستخدم المخطط الشريطي Bar Chart أو النقطي Dot Chart. سنرى كيف بإمكاننا رسم هذه الرسوم البيانية باستخدام seaborn وبيانات الناجين من سفينة تايتانيك:

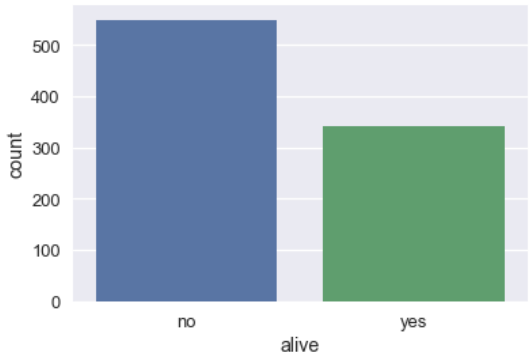
```
import seaborn as sns
sns.set()

# تحميل البيانات
ti = sns.load_dataset('titanic').reset_index(drop=True)

df_interact(ti)
```

```
(891 rows, 15 columns) total
```

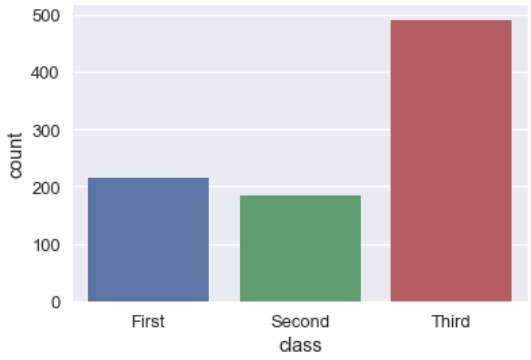
المخطط الشريطي

في seaborn، يوجد نوعين من المخططات الشريطية. النوع الأول يستخدم دالة countplot ليقوم بعد عدد مرات تكرار كل نوع في العمود:

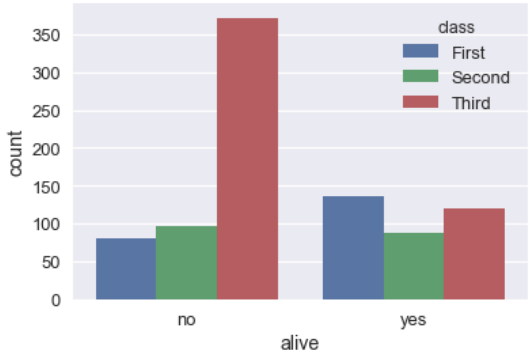
```
# يقوم برسم وحساب عدد الناجين والذين لم ينجو ثم رسم شريط بعددهم
sns.countplot(x='alive', data=ti);
```




```
sns.countplot(x='class', data=ti);
```

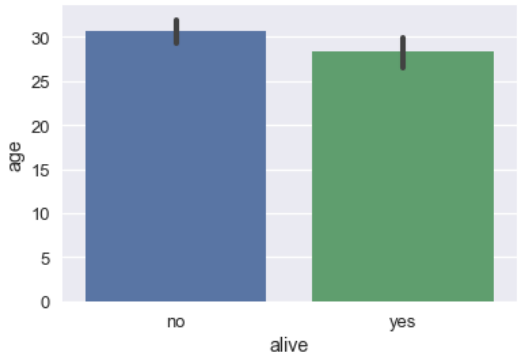



```
# كما في مخطط الصندوق، يمكننا التقسيم بشكل اوسع باستخدام الألوان
sns.countplot(x='alive', hue='class', data=ti);
```



والطريقة الأخرى، دالة barplot، تجمع البيانات باستخدام عمود نوعي ويتم تحديد طول الشريط بِنَاءًا على متوسط نتيجة عمود كمي لكل نوع من الأنواع.

```
# لكل نوع ناجي/لم ينجو، قم بحساب ورسم متوسط الأعمار
sns.barplot(x='alive', y='age', data=ti);
```



يمكننا حساب طول كل شريط عن طريق جمع الأعمدة alive و age وحساب المتوسط للعمود age:

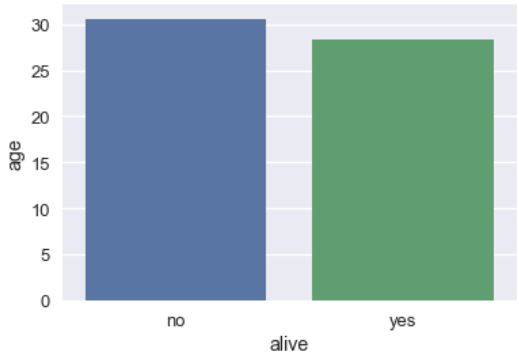
```
ti[['alive', 'age']].groupby('alive').mean()
```

	Age
alive	
no	30.626179
yes	28.343690

بشكل تلقائي، دالة barplot تحاول حساب فاصل ثقة بنسبة 95% باستخدام Bootstrap لكل متوسط. مشار إليها بالخط الأسود في أعلى الشريط. فاصل الثقه يحاول توضيح أنه إذا كانت بيانات تايتانيك تحتوي على عينة عشوائية، الفرق في عمر الراكب بين الذين نجو والذين لم ينجو ليس مهم إحصائياً على مستوى 5%.

فواصل الثقة تأخذ وقتاً طويلاً للحساب عندما يكون لدينا بيانات ذات حجم أكبر، لذا من الأفضل عدم حسابها:

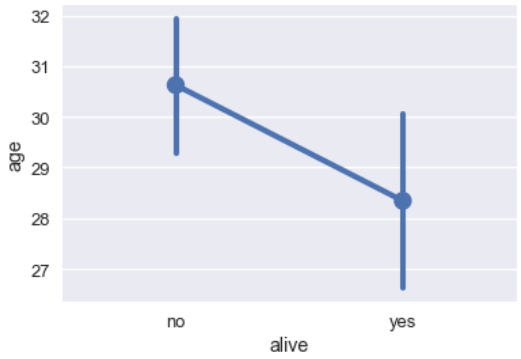
```
sns.barplot(x='alive', y='age', data=ti, ci=False);
```



المخطط النقطي

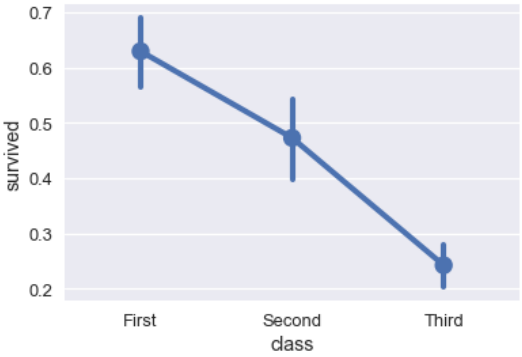
المخطط النقطي يتشابه كثيراً مع المخطط الشريطي. بدلاً من رسم أشرطة، المخطط النقطي يرسم نقطة واحدة تعبر عن نهاية الشريط. نستخدم الدالة pointplot في seaborn لرسم المخطط النقطي. كما في barplot، دالة pointplot تقوم بالتجميع أوتوماتيكياً وحساب المتوسط لعمود رقمي، وتحديد فاصل ثقة بنفسه 95% بخطوط عاموديه يكون النقطة متوسطها:

```
# لكل نوع ناجي/لم ينجو، قم بحساب ورسم متوسط الأعمار
sns.pointplot(x='alive', y='age', data=ti);
```

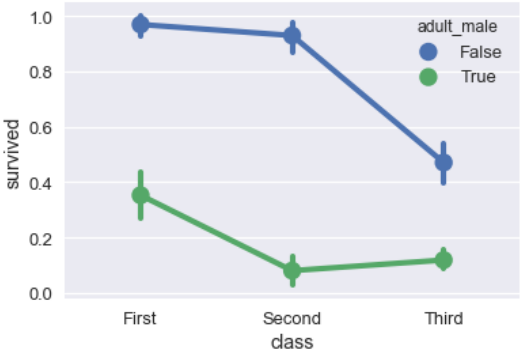


المخطط النقطي مفيد جداً عند محاولة مقارنة التغيرات على حسب نوع البيانات:

```
# عرض نسبة الناجين بناءا على الفئة
sns.pointplot(x='class', y='survived', data=ti);
```




```
# عرض نسبة الناجين بناءا على الفئة
# التقسيم ما اذا كان الناجي ذكر بالغ
sns.pointplot(x='class', y='survived', hue='adult_male', data=ti);
```



تخصيص الرسوم البيانية

تخصيص الرسوم البيانية باستخدام matplotlib

على الرغم ان seaborn تُمكنا من رسم العديد من أنواع الرسوم البيانية بشكل سريع، لكنها لا يمكننا التحكم بشكل كامل بالرسم. مثلاً، لا يمكننا في seaborn تعديل عنوان الرسم البياني، وعناوين محاور الطول والعرض، أو إضافة ملاحظات على الرسم. لذاء يجب علينا استخدام مكتبة matplotlib والتي تعتمد عليها seaborn

مكتبة seaborn هي بالأساس matplotlib ولكن أضيفت لها تحسينات وتعديلات لتتمكن من التعامل مع مكتبة pandas.

توفر لنا matplotlib الأساسيات لإنشاء الرسوم البيانية في بايثون. على الرغم من أنها تمكنا من الحصول على تحكم كامل، إلا أنها أيضاً متعبة في العمل، إعادة رسم ما رسمناه سابقاً في seaborn باستخدام matplotlib سنحتاج لكتابة العديد من الأسطر البرمجية. بالأصح، يمكننا القول إن seaborn هي اختصار مفيد لإنشاء رسوم بيانية في matplotlib. نفضل العمل بالرسوم المبدئية المقدمة من seaborn، ولكن لتعديل وتخصيص الرسومات لغرض النشر سنحتاج لمعرفة أساسيات العمل في matplotlib.

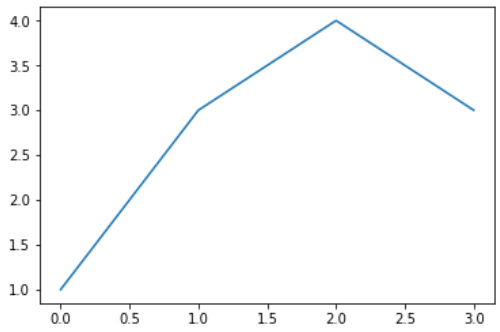
قبل أن نرى المثال الأول، يجب علينا تفعيل دعم مكتبة matplotlib في Jupyter Notebook.

```
# هذا السطر يجعل الرسوم البيانية تظهر كصور في جوبتر بدلاً من ان تظهر في صفحة منفصلة
%matplotlib inline

# matplotlib هو اختصار كثير الأستخدام لمكتبة plt الاختصار
import matplotlib.pyplot as plt
```

تخصيص الرسم والأبعاد

للرسم في matplotlib، نحتاج أولاً لإنشاء شكل بياني أو *Figure*، ثم إضافة المحاور *Axes* للشكل. في matplotlib، المحور *Axes* هو رسم بياني واحد، والشكل البياني *Figure* يمكن أن يحتوي على عدة محاور على شكل جداول. المحور يحتوي على نقاط *Marks*، وهي الخطوط أو النقاط المرسومة في الرسم البياني:

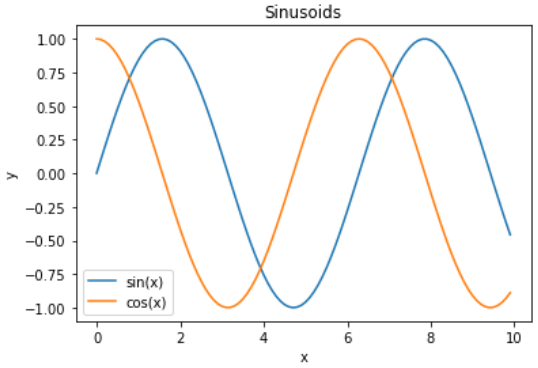
```
# انشاء الشكل البياني
f = plt.figure()

# اضافة محور إلى الشكل البياني. المتغير الثاني والثالث تقوم بانشاء
# جدول بسطر وعمود واحد. المتغير الاول يتحكم بمكان المحور، هنا في الخليه الأولى
ax = f.add_subplot(1, 1, 1)

# انشاء رسم بياني خطي على المحور
ax.plot([0, 1, 2, 3], [1, 3, 4, 3])

# أظهار الرسم البياني، في جوبتر يتم نداء هذه الدالة بشكل اوتوماتيكي
# لذا لن نستخدمها في الأكواد البرمجية القادمة
plt.show()
```



للتخصيص في هذه الرسمه، يمكننا استخدام دوال أخرى على كائن المحور:

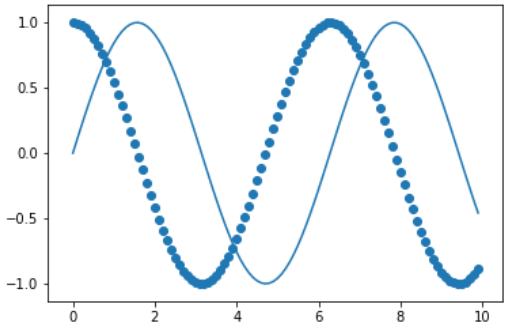
كائن هنا تعني Object

```
f = plt.figure()
ax = f.add_subplot(1, 1, 1)

x = np.arange(0, 10, 0.1)

# تمكننا من انشاء عنوان للخط label اضافة المتغير
ax.plot(x, np.sin(x), label='sin(x)')
ax.plot(x, np.cos(x), label='cos(x)')
ax.legend()

ax.set_title('Sinusoids')
ax.set_xlabel('x')
ax.set_ylabel('y');
```



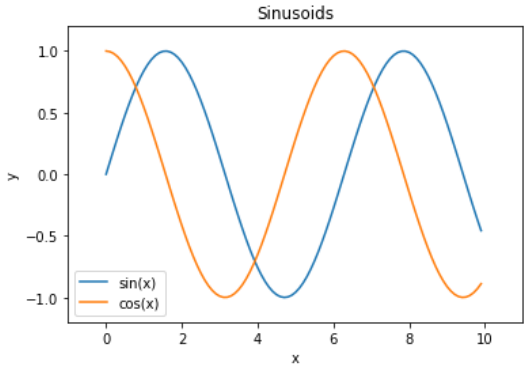
كاختصار، matplotlib لديها دوال رسم بياني في plt يمكنها إنشاء شكل بياني ومحور أوتوماتيكياً:

```
# اختصار لرسم شكل بياني ومحور
plt.plot(x, np.sin(x))

# اكثر من مره في نفس الخليه، سيتم إعادة استخدام نفس الشكل والمحور plt عند استخدام
plt.scatter(x, np.cos(x));
```



يحتوي plt على دوال مشابهة للتي في المحور، لذا يمكننا إعادة إنشاء أحد الرسوم في الأعلى باستخدام plt مباشره:

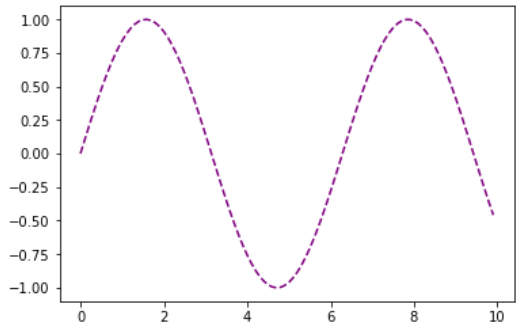
```
x = np.arange(0, 10, 0.1)

plt.plot(x, np.sin(x), label='sin(x)')
plt.plot(x, np.cos(x), label='cos(x)')
plt.legend()

# اختصار ل ax.set_title
plt.title('Sinusoids')
plt.xlabel('x')
plt.ylabel('y')

# تحديد مقاسات محوري x و y
plt.xlim(-1, 11)
plt.ylim(-1.2, 1.2);
```



تخصيص النقاط

لتغير خصائص النقاط في الرسمه (مثلاً، الخطوط في الرسمه السابقة)، يمكننا تمرير المزيد من المتغيرات إلى plt.plot:

```
plt.plot(x, np.sin(x), linestyle='--', color='purple');
```



تصفح الشرح الخاص بمكتبة matplotlib هو أسهل طريقة لمعرفة أي المتغيرات متوفرة لكل دالة. الطريقة الأخرى هي حفظ الكائن الناتج عن دالة الرسم:

```
line = plt.plot([1,2,3])
```

كائن الخط في الرسم البياني يحتوي على الكثير من الخصائص التي يمكن أن تتحكم فيها، يمكن عرضها باستخدام .set، هذه قائمة كاملة بخصائص الرسم البياني السابق:

```
line.set
```

```
line.set                    line.set_drawstyle          line.set_mec
line.set_aa                 line.set_figure             line.set_mew
line.set_agg_filter         line.set_fillstyle          line.set_mfc
line.set_alpha              line.set_gid                line.set_mfcalt
line.set_animated           line.set_label              line.set_ms
line.set_antialiased        line.set_linestyle          line.set_picker
line.set_axes               line.set_linewidth          line.set_pickradius
line.set_c                  line.set_lod                line.set_rasterized
line.set_clip_box           line.set_ls                 line.set_snap
line.set_clip_on            line.set_lw                 line.set_solid_capstyle
line.set_clip_path          line.set_marker             line.set_solid_joinstyle
line.set_color              line.set_markeredgecolor    line.set_transform
line.set_contains           line.set_markeredgewidth    line.set_url
line.set_dash_capstyle      line.set_markerfacecolor    line.set_visible
line.set_dashes             line.set_markerfacecoloralt line.set_xdata
line.set_dash_joinstyle     line.set_markersize         line.set_ydata
line.set_data               line.set_markevery          line.set_zorder
```

ولكن استخدام setp (اختصار لتحديد خاصية) قد يكون أفضل، خاصة عند العمل بشكل تفاعلي كونه يحتوي على وصف، لكي تتعلم الطرق الصحيحة لنداء الدوال أثناء عملك عليها:

```
line = plt.plot([1,2,3])
```

```
plt.setp(line, 'linestyle')
```

```
  linestyle: {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
```

```
plt.setp(line)
```

```
agg_filter: unknown
alpha: float (0.0 transparent through 1.0 opaque)
animated: [True | False]
```

```
antialiased or aa: [True | False]
...
...
...
```

في النتيجة الأولى يظهر لنا القيم الصحيحة التي يقبلها متغير linestyle، وفي النتيجة الثانية يظهر لنا جميع المتغيرات التي يمكن تخصيصها في كائن الخط. يمكنك ذلك بسهولة اكتشفت وتخصيص رسومك البيانية لتحصل على ما تريده.

النصوص ودعم LaTeX

في matplotlib يمكن إضافة النص لمحور معين أو للشكل البياني ككل.

هذه الأوامر لإضافة نصوص للمحاور:

- set_title: إضافة عنوان
- set_xlabel: إضافة عنوان للمحور السيني x-axis
- set_ylabel: إضافة عنوان للمحور الصادي y-axis
- text: إضافة نص في مكان معين
- annotate: إضافة ملاحظة مع إمكانية إضافة سهم

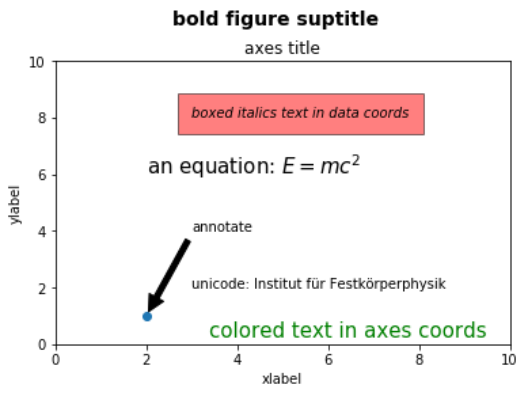
والأوامر التالية للشكل البياني:

- figtext: إضافة نص في مكان معين
- suptitle: إضافة عنوان

وأي نص من الممكن أن يحتوي على نصوص بشكل LaTeX والمخصصة للعمليات الرياضية، طالما تم إضافتها بداخل علامتي $.

هذا المثال على جميع ما سبق:

```
fig = plt.figure()
fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')

ax = fig.add_subplot(1, 1, 1)
fig.subplots_adjust(top=0.85)
ax.set_title('axes title')

ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')

ax.text(3, 8, 'boxed italics text in data coords', style='italic',
        bbox={'facecolor':'red', 'alpha':0.5, 'pad':10})

ax.text(2, 6, 'an equation: $E=mc^2$', fontsize=15)

ax.text(3, 2, 'unicode: Institut für Festkörperphysik')

ax.text(0.95, 0.01, 'colored text in axes coords',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)


ax.plot([2], [1], 'o')
ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.axis([0, 10, 0, 10]);
```



تخصيص رسوم seaborn باستخدام matplotlib

والآن بعد أن رأينا كيف نستخدم matplotlib لتخصيص الرسم البياني، يمكننا استخدام نفس الدوال للتخصيص في seaborn لأن seaborn تقوم بإنشاء الرسوم البيانية باستخدام matplotlib:

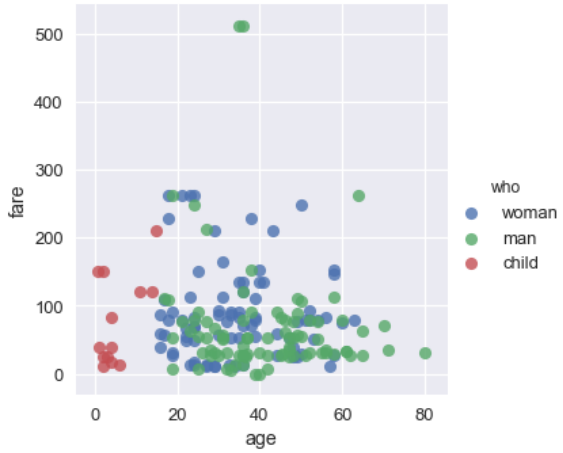
```
# تحميل مكتبة seaborn
import seaborn as sns
sns.set()

# ضبط شكل وحجم الرسم
# https://seaborn.pydata.org/generated/seaborn.set_context.html
sns.set_context('talk')

# تحميل البيانات
ti = sns.load_dataset('titanic').dropna().reset_index(drop=True)
ti.head()
```

	survived	pclass	sex	age	...	deck	embark_town	alive	alone
0	1	1	female	38	...	C	Cherbourg	yes	FALSE
1	1	1	female	35	...	C	Southampton	yes	FALSE
2	0	1	male	54	...	E	Southampton	no	TRUE
3	1	3	female	4	...	G	Southampton	yes	FALSE
4	1	1	female	58	...	C	Southampton	yes	TRUE

5 rows × 15 columns

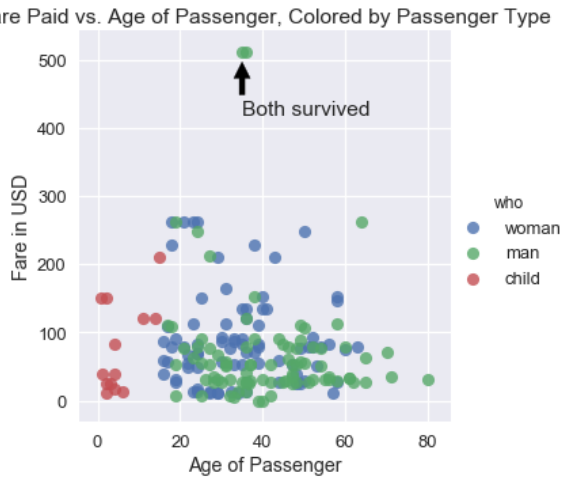
لنبدأ بهذه الرسمه:

```
sns.lmplot(x='age', y='fare', hue='who', data=ti, fit_reg=False);
```



يمكننا رؤية أن الرسم يحتاج لعنوان وصف أفضل لعناوين المحورين x و y. أيضاً، الشخصين اللذين لديهما أعلى التذاكر سعراً نجا، لذا نحتاج لإضافة توضيح ذلك في رسمنا البياني:

```
sns.lmplot(x='age', y='fare', hue='who', data=ti, fit_reg=False)

plt.title('Fare Paid vs. Age of Passenger, Colored by Passenger Type')
plt.xlabel('Age of Passenger')
plt.ylabel('Fare in USD')

plt.annotate('Both survived', xy=(35, 500), xytext=(35, 420),
             arrowprops=dict(facecolor='black', shrink=0.05));
```



بشكل عملي، سنستخدم `seaborn` لتصفح البيانات بشكل سريع ثم ننتقل إلى `matplotlib` لنحسن من الرسوم بعد اختيارنا للرسم البياني المثالي.

مبادئ تصوير البيانات

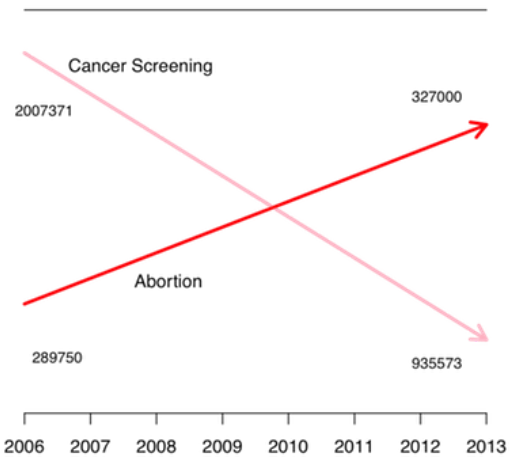
والآن بما أن لدينا الأدوات لرسم وتخصيص الرسوم البيانية، ننتقل للمبادئ الأساسية لتصوير البيانات. تماماً كما في باقي أقسام علم البيانات، من الصعب تحديد مقاييس معينة لكفاءة رسمه بيانية معينة. لكن، هناك مبادئ أساسيه وجودها يجعل الرسوم البيانية فعالة بشكل كبير ويظهر اتجاهات البيانات. سنناقش ست مبادئ: المقياس، الحالة، الإدراك، التحويل، النصوص والتبسيط.

مبادئ المقاييس

يقصد بمبادئ المقاييس هو اختيار المقاسات المناسبة للمحاور x و y في الرسم البياني.

في جلسة استماع في الكونجرس الأمريكي عام 2015، الوكيل شافيتز ناقش تحقيق عن برنامج Planned Parenthood. قام الوكيل باستخدام الرسم البياني التالي والذي ظهر في تقرير من قبل مؤسسة Americans United for Life. يقارن الرسم بين اجراءات الإجهاض و فحوص السرطان، كلا الإجرائين يتم تقديمها في Planned Parenthood. (التقرير متوفر بشكل كامل هنا).

ما المريب بهذا الرسم البياني؟ كم نقطة تم رسمها؟



هذا الرسم يخالف مبادئ المقاييس، لا يحسن الاختيار لمحوري x و y.

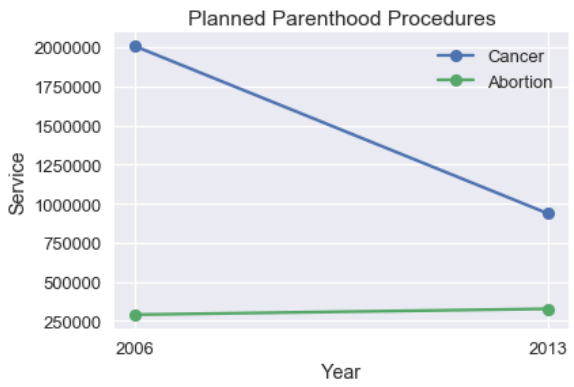
عندما نختار المحورين x و y في رسمنا البياني، يجب أن نبقي مقاساتنا ثابتة على طول المحور. ولكن، الرسمه العلوية تحتوي على مقاسين مختلفين لخطي إجراءات الإجهاض وفحوص السرطان، بداية خط إجراءات الإجهاض ونهاية خط فحوص السرطان يبدوان قريبين من بعضهما على المحور y ولكن الأرقام تظهر العكس. أيضاً، تم رسم النقاط لسنتي 2006 و 2013 ولكن المحور x يحتوي على علامات إضافية لسنوات بينهما ولم يتم استخدامها.

للتحسين من هذا الرسم البياني، يجب علينا إعادة رسم النقاط على مقاس ثابت في المحور y:

لتحميل بيانات Planned Parenthood اضغط هنا.

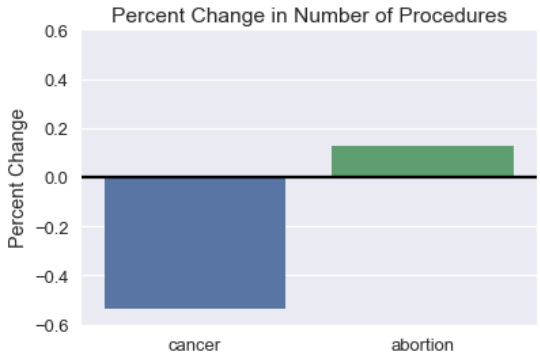
```
pp = pd.read_csv("data/plannedparenthood.csv")
```

```
plt.plot(pp['year'], pp['screening'], linestyle="solid", marker="o", label='Cancer')
plt.plot(pp['year'], pp['abortion'], linestyle="solid", marker="o", label='Abortion')
plt.title('Planned Parenthood Procedures')
plt.xlabel("Year")
plt.ylabel("Service")


plt.xticks([2006, 2013])
plt.legend();
```

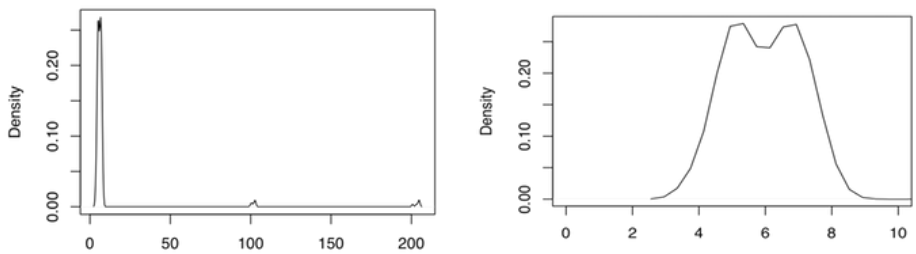


يمكن أن نلاحظ أن الفرق في إجراءات الإجهاض قليل مقارنة بالنزول المفاجئ في فحوص السرطان. بدلاً من مقارنة عدد الإجراءات بينهما، ربما من الأفضل مقارنة فرق بالنسبة بين السنتين 2006 و 2013:

```
percent_change = pd.DataFrame({
    'percent_change': [
        pp['screening'].iloc[1] / pp['screening'].iloc[0] - 1,
        pp['abortion'].iloc[1] / pp['abortion'].iloc[0] - 1,
    ],
    'procedure': ['cancer', 'abortion'],
    'type': ['percent_change', 'percent_change'],
})
ax = sns.barplot(x='procedure', y='percent_change', data=percent_change)
plt.title('Percent Change in Number of Procedures')
plt.xlabel('')
plt.ylabel('Percent Change')
plt.ylim(-0.6, 0.6)
plt.axhline(y=0, c='black');
```



عند اختيار مقاسات المحاور نحاول التركيز على الجزء الذي يحتوي على عدد بيانات أكبر، خاصة عندما نعمل على بيانات المعروفة بأسم الذيل الطويل Long-tailed Data. لاحظ في الرسم البياني التالي الرسمه قبل (يسار) وبعد (يمين) التقريب في على الرسم الأخرى: 📝



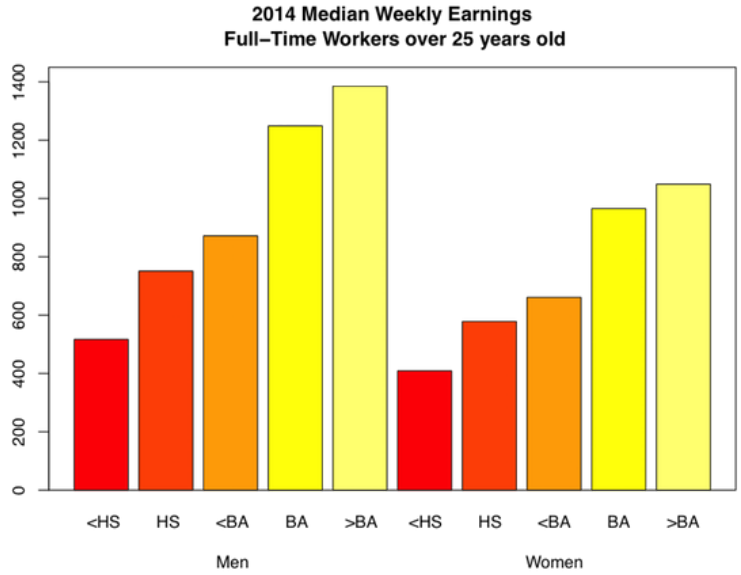
الرسمه على اليمين تفيد أكثر في فهم البيانات. إذا احتاج الأمر، يمكننا رسم أكثر من رسمه بيانيه لأجزاء مختلفة من البيانات. لاحقاً في هذا الفصل، سنتحدث عن تحويل البيانات والذي سيساعدنا في رسم البيانات ذات الذيل الطويل.

مبادئ الحالات

تقدم لنا مبادئ الحالات الطرق والوسائل لإظهار التوزيع والعلاقة بين المجموعات الفرعية في بياناتنا.

يشرف مكتب إحصائيات العمال في الولايات المتحدة على إجراء استبيانات علمية لصحة اقتصاد الولايات المتحدة. يحتوي موقع المكتب على أداة لإنشاء التقارير باستخدام هذه البيانات التي استخدمناها لرسم متوسط الأجر الأسبوعي حسب الجنس.

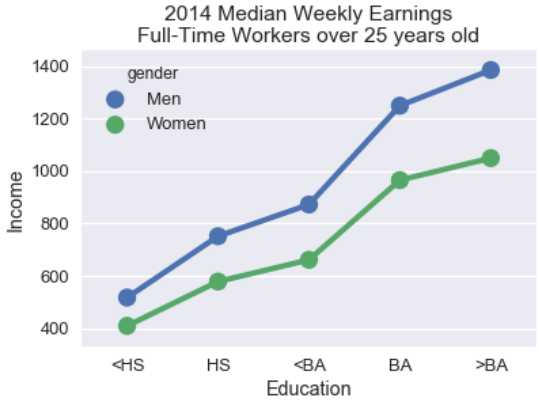
أي المقارنات أسهل في هذا الرسم؟ هل هذه المقارنات هي الأهم؟



يبين لنا هذا الرسم لمحه بسيطه على أن الراتب الأجر الأسبوعي يزيد حسب الدرجة العلمية. ولكن، من الصعب تحديد الزيادة بالضبط بين كل درجه علميه وأخرى ويصعب أكثر التفريق بين أجور الذكور والإناث في نفس الدرجة العلمية. يمكننا التحقق من اتجاهات المشكلتين السابقتين باستخدام المخطط النقطي بدلاً من الشريطي:

لتحميل بيانات الأجور اضغط هنا.

```
cps = pd.read_csv("data/edInc2.csv")
ax = sns.pointplot(x="educ", y="income", hue="gender", data=cps)

ticks = ["<HS", "HS", "<BA", "BA", ">BA"]
ax.set_xticklabels(ticks)
ax.set_xlabel("Education")
ax.set_ylabel("Income")
ax.set_title("2014 Median Weekly Earnings\nFull-Time Workers over 25 years old");
```

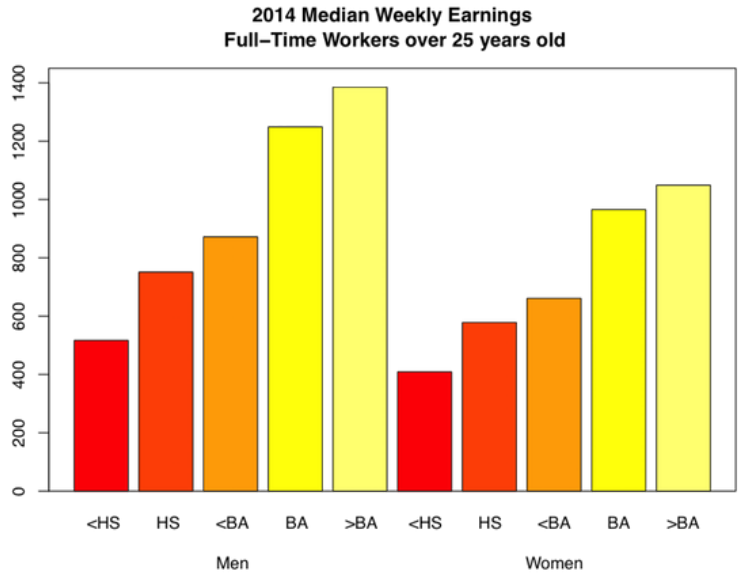


الخط الذي يربط بين النقاط يوضح تأثير الحصول على درجة البكالوريوس على الأجر الأسبوعي. وضع نقطتي الذكور والإناث فوق بعضهما يسهل علينا ملاحظة زيادة الفارق بينهما كلما زادت الدرجة العلمية.

للمساعدة في المقارنة بين المجموعتين الفرعيتين في بياناتنا، حاذي النقاط على المحور x أو y واختر ألوان أو أشكال مختلفة لكل منهما. الخطوط عادةً ما توضح اتجاه البيانات بشكل واضح أكثر من المخططات الشريطية وهي اختيار أفضل لنوعي البيانات النوعية والكمية.

مبادئ الإدراك

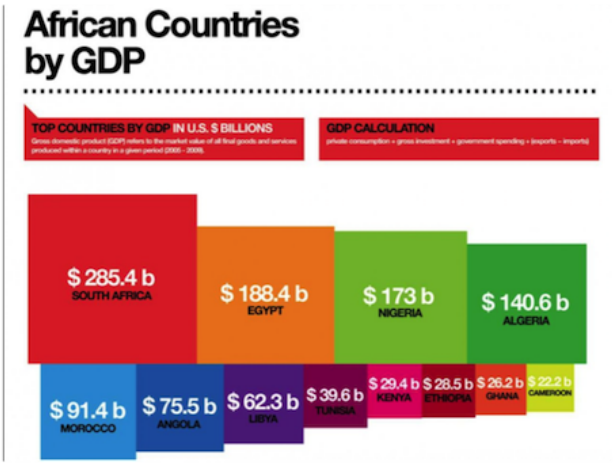
لدى إدراك البشر بعض الخصائص التي يجب ان ننتبه لها عند تصميم رسومنا البيانية. أول الخصائص في الإدراك البشري أننا ننتبه لبعض الألوان أكثر من أخرى، خاصة في درجات اللون الأخضر. أيضاً، نلاحظ أن الألوان الفاتحة أكبر حجماً من الألوان الغامقة. مثلاً، في الرسم البياني للأجور الأسبوعية، المخططات الشريطية الفاتحة تجذب انتباهنا أكثر من الغامقة:



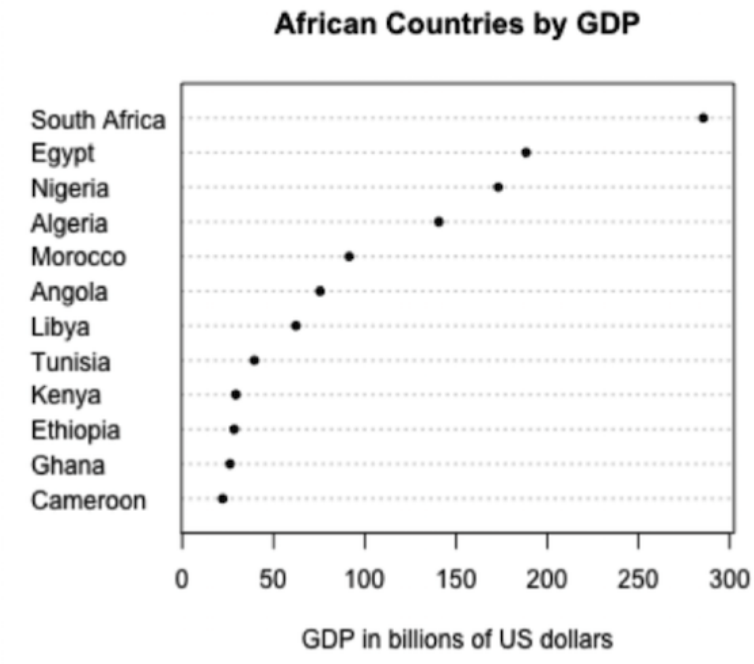
عملياً، يجب أن تتأكد أن ألوان رسومك البيانية ذات إدراك موحد *Perceptually Uniform*. يعني ذلك، شدة الألوان لا تتغير بين كل شريط وآخر في الرسم. في البيانات الكمية، لدينا خياران: إذا كانت البيانات من تتصاعد من الأقل للأعلى فيجب عليك أن تشدد على القيم العليا، لذا تستخدم قالب ألوان متسلسلة *Sequential Color Scheme* والتي تعين لوناً فاتحاً للقيم العالية. إذا كانت كلا القيم العليا والدنيا تحتاج للتنويه عليها، استخدم قالب ألوان معكوس *Diverging Color Scheme* فيه يتم تعيين اللون الفاتح للقيم القريبة من المنتصف.

تحتوي seaborn على الكثير من قوالب الألوان. يمكنك تصفحها في شرح المكتبة لمعرفة كيفية التغير بين كل قالب وآخر:
http://seaborn.pydata.org/tutorial/color_palettes.html

ثاني خصائص الإدراك البشري هي أننا دقيقين أكثر عندما نقارن بين الأطوال وأقل دقه عندما نقارن بين المساحات. الرسم البياني التالي للنواتج المحلية الإجمالية لبلدان أفريقيا:

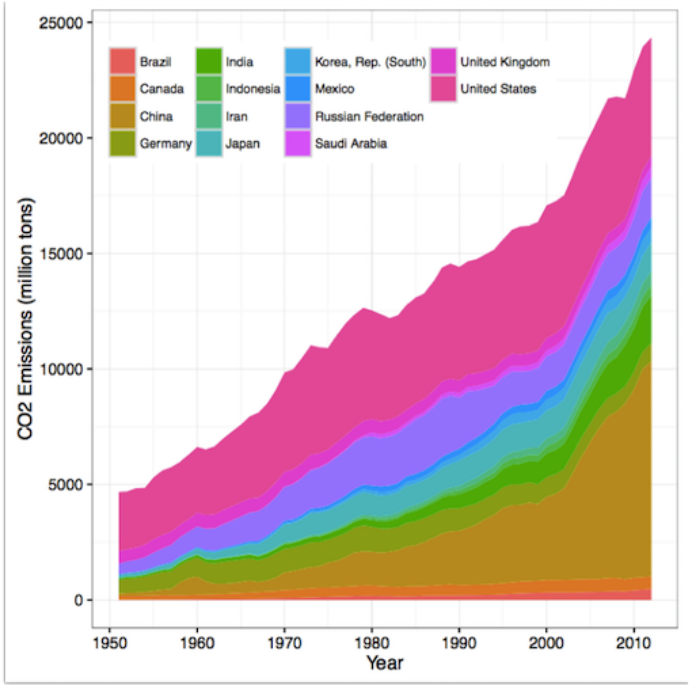


بِنَاءًا على الأرقام، جنوب أفريقيا لديها ضعف الناتج المحلي في بالجزائر ولكن ليس من السهل ملاحظة ذلك في الرسم البياني. بدلاً من ذلك، يمكننا استخدام المخطط النقطي للرسم:



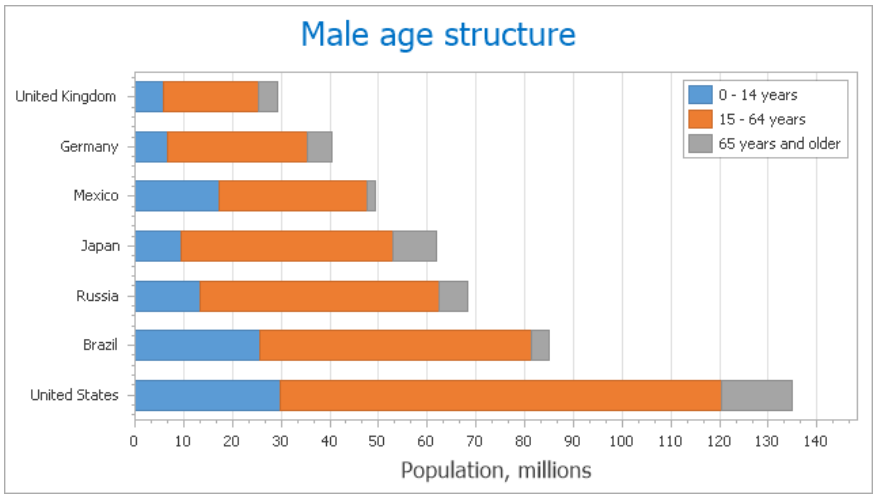
هذا أوضح بكثير لأنه سمح لنا المقارنة الطول بدلاً من المساحة. الدائرة الجزئية Pie Chart والرسوم ثلاثية الأبعاد Three-dimensional Charts يصعب ترجمتها بسبب نفس المشكلة؛ عملياً نحاول الابتعاد عن هذه الرسومات.

خاصية الإدراك البشري الثالثة والأخيرة هي أن العين تواجه صعوبات عند التنقل بين خطوط الأساس. في الرسم البياني التالي من النوع المساحي المُجَمع (المتكدس) Stacked Area Chart والذي يوضح انبعاثات ثاني أكسيد الكربون بمرور الوقت ومفصله حسب الدولة:



من الصعب معرفة ما إذا زادت أو نقصت الانبعاثات في المملكة المتحدة بسبب مشكلة هزهزة خط الأساس Jiggling Baseline: خط الأساس (الخط السفلي) للمساحة يهتز أعلى وأسفل. أيضا من الصعب المقارنة بين انبعاثات المملكة المتحدة والصين عندما يكون الطول متشابه (في عام 2000 مثلاً).

نفس المشكلة تظهر في المخطط الشريطي المُجَمع (المتكدس). في الرسم البياني التالي يصعب المقارنة عدد الذين أعمارهم بين 15-64 في المكسيك وألمانيا.

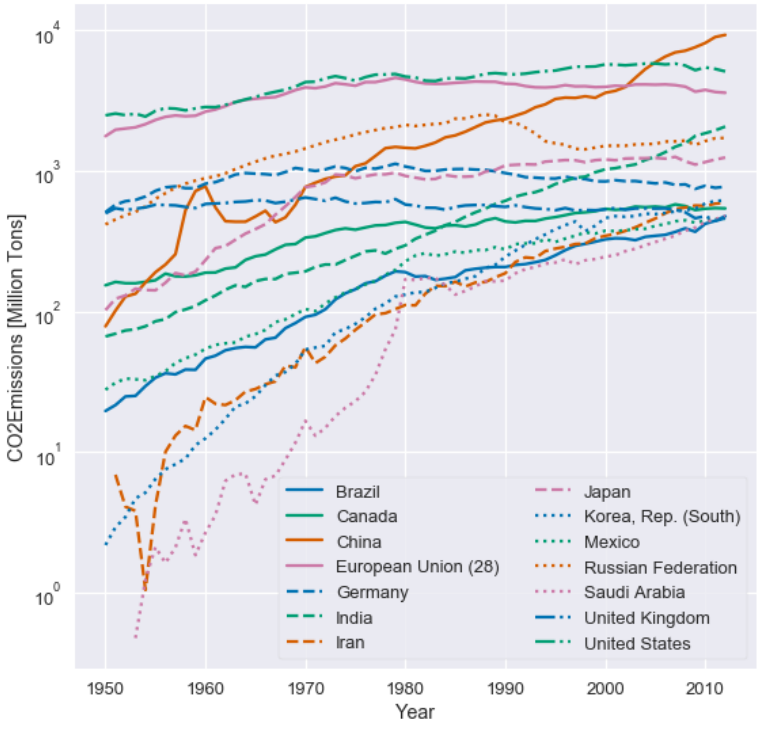


يمكننا تحسين رسوم البيانات من النوع المساحي والشريطي المُجمعه (متكدسة) عن طريق تحويلها إلى رسوم خطيه. الرسم البياني التالي لبيانات انبعاثات ثاني اكسيد الكربون بمرور الوقت كخطوط بدلاً من مساحات:

لتحميل بيانات انبعاثات ثاني أكسيد الكربون بين الدول اضغط هنا.

```
co2 = pd.read_csv("data/CAITcountryCO2.csv", skiprows = 2,
                  names = ["Country", "Year", "CO2"])
last_year = co2.Year.iloc[-1]
q = f"Country != 'World' and Country != 'European Union (15)' and Year == {last_year}"
top14_lasty = co2.query(q).sort_values('CO2', ascending=False).iloc[:14]
top14 = co2[co2.Country.isin(top14_lasty.Country) & (co2.Year >= 1950)]

from cycler import cycler

linestyles = (['-', '--', ':', '-.']*3)[:7]
colors = sns.color_palette('colorblind')[:4]
lines_c = cycler('linestyle', linestyles)
color_c = cycler('color', colors)

fig, ax = plt.subplots(figsize=(9, 9))
ax.set_prop_cycle(lines_c * color_c)
```

```
x, y ='Year', 'CO2'
for name, df in top14.groupby('Country'):
    ax.semilogy(df[x], df[y], label=name)

ax.set_xlabel(x)
ax.set_ylabel(y + "Emissions [Million Tons]")
ax.legend(ncol=2, frameon=True);
```



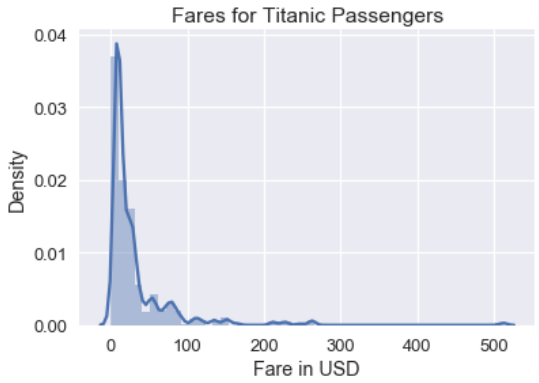
رسم البيانات على شكل خطوط لا يحدث بها اهتزازات في خط الأساس لكل خط لذا من السهل مقارنة الانبعاثات بين الدول. أيضا يمكننا أن نرى بوضوح أي الدول زادت فيها الانبعاثات.

مبادئ التحويل

تقدم لنا مبادئ التحويل طُرق مفيدة لتغير البيانات في الرسوم البيانية لإظهار اتجاهاتها بصورة واضحة. عادة ما نستخدم التحويل لإظهار الأنماط في البيانات المنحرفة Skewed Data أو العلاقات غير الخطية بين المتغيرات.

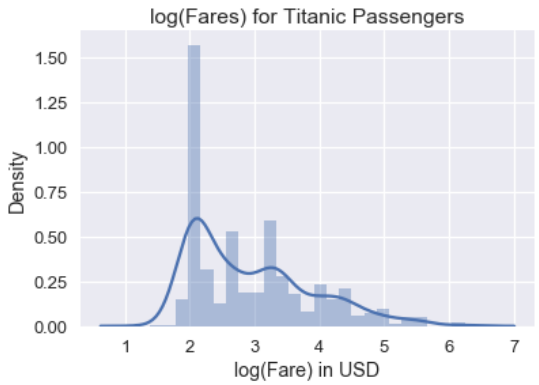
الرسم البياني التالي يوضح توزيع أسعار التذاكر بين كل راكب في سفينة تايتانيك. كما ترى،الرسم منحرف باتجاه اليمين:

```
ti = sns.load_dataset('titanic')
sns.distplot(ti['fare'])
plt.title('Fares for Titanic Passengers')
plt.xlabel('Fare in USD')
plt.ylabel('Density');
```



على الرغم أن الرسم البياني يوضح جميع أسعار التذاكر، من الصعب ملاحظة الأنماط فيها لأن البيانات مُتجمعة في الجهة اليسرى. لتحسين ذلك، يمكننا حساب اللوغاريتم الطبيعي Natural logarithm لأسعار التذاكر قبل رسمها:

```
sns.distplot(np.log(ti.loc[ti['fare'] > 0, 'fare']), bins=25)
plt.title('log(Fares) for Titanic Passengers')
plt.xlabel('log(Fare) in USD')
plt.ylabel('Density');
```



يمكننا ملاحظة في الرسم البياني أنه بعد حساب اللوغاريتمية أكثر القيم تكراراً (المنوال Mode) كان حوالي $e^2 = \$7.40$ والأقل تكرارا كان $e^{3.4} = \$30.00$. لماذا يفيدنا رسم اللوغاريتمية الطبيعية للبيانات عندما تكون مُنحرفة؟ اللوغاريتمية للأرقام العاليه عادة ما تكون قريبة من اللوغاريتمية للأرقام الصغيرة:

value	(log(value
1	0
10	2.3
50	3.91
100	4.6
500	6.21
1000	6.9

يعني ذلك أن أخذ اللوغارتمية ستجلب تلك البيانات ذات الأرقام الكبيرة على الجانب الأيمن إلى الأرقام الصغيرة. يُسهل ذلك في توضيح أين تقع أغلب البيانات.

بالأصح، تعتبر اللوغارتم أداه مهمة في تحويل البيانات، تساعدنا أيضاً على ملاحظة العلاقات بين البيانات غير الخطية. في عام 1619، وثق يوهانس كيبلر البيانات التالية ليكتشف القانون الثالث في حركة الكواكب:

لتحميل بيانات يوهانس كيبلر عن حركة الكواكب اضغط هنا.

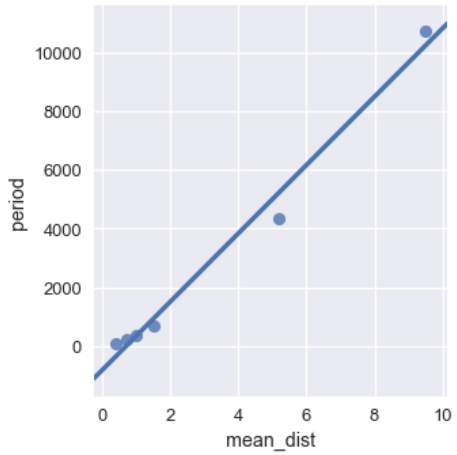
```
planets = pd.read_csv("data/planets.data", delim_whitespace=True,
                      comment="#", usecols=[0, 1, 2])
planets
```

	planet	mean_dist	period
0	Mercury	0.389	87.77
1	Venus	0.724	224.7
2	Earth	1	365.25
3	Mars	1.524	686.95
4	Jupiter	5.2	4332.62
5	Saturn	9.51	10759.2

إذا رسمنا متوسط المسافة حتى الشمس مقارنة بفترة الدوران، يمكننا ملاحظة أنه هناك علاقة لا تبدو خطيه:

```
sns.lmplot(x='mean_dist', y='period', data=planets, ci=False)
```

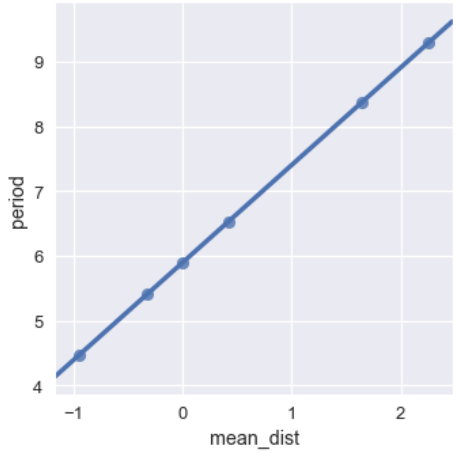
```
<seaborn.axisgrid.FacetGrid at 0x1a1f54aba8>
```



ولكن، إذا قمنا بحساب اللوغاريتمية لمتوسط المسافة والفترة، سنحصل على الرسم التالي:

```
sns.lmplot(x='mean_dist', y='period',
           data=np.log(planets.iloc[:, [1, 2]]),
           ci=False);
```

```
<seaborn.axisgrid.FacetGrid at 0x1a1f693da0>
```



يمكن أن نرى علاقة خطية Linear بشكل شبه كامل هنا بين القيم اللوغاريتمية للمتوسط والفترة. ماذا يعني ذلك؟ بما أننا نعتقد أن هناك علاقة بين القيم اللوغارتمية، يمكننا استنتاج التالي:

$$
\begin{aligned}
\log(period) &= m\log(dist) + b \\
period &= e^{m\log(dist)+b} \qquad \text{Taking the exponent of both sides} \\
period &= e^b dist^m \\
period &= C \cdot dist^m
\end{aligned}
$$

قمنا بتبديل e^b ب C في الخطوة الأخيرة لتوضيح أن e^b رقم ثابت. العملية الرياضية الجبرية السابقة تظهر أنه عندما تكون قيمتان لديهما علاقة متعددة الحدود Polynomial ، فإن لوغارتميتهما ذات علاقة خطيه. بالأصح، يمكننا إيجاد درجة متعددة حدود عن طريق إيجاد انحدار الخط Slope. في هذه الحالة، الانحدار قيمته 1.5 والذي يعطينا قانون كيبلر الثالث: $period \propto dist^{1.5}$

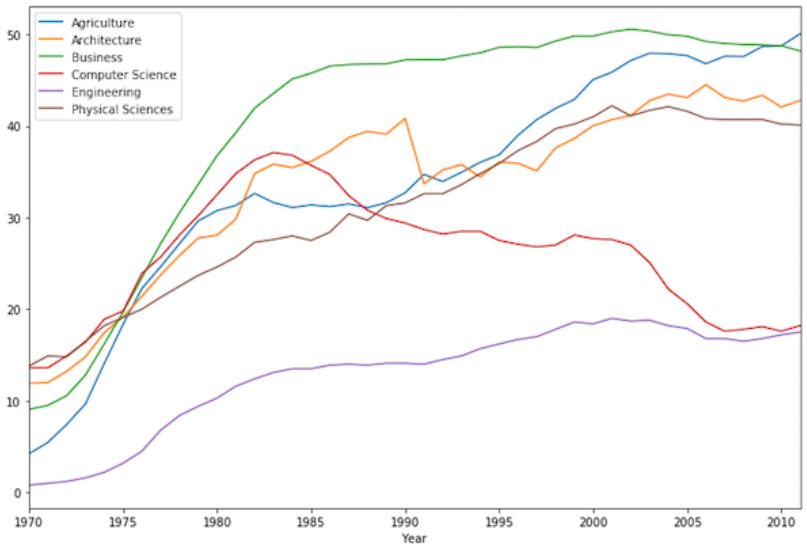
بنفس طريقة الاستنتاج السابق يمكننا إظهار أن العلاقة بين $\log(y)$ و x خطيه، القيمتين لديهما علاقة أسيه: $y = a^x$

بهذه الطرق، يمكننا استخدام اللوغارتمية لإظهار أنماط في البيانات ذات الذيل الطويل وعلاقات الغير خطية الشائعة بين القيم.

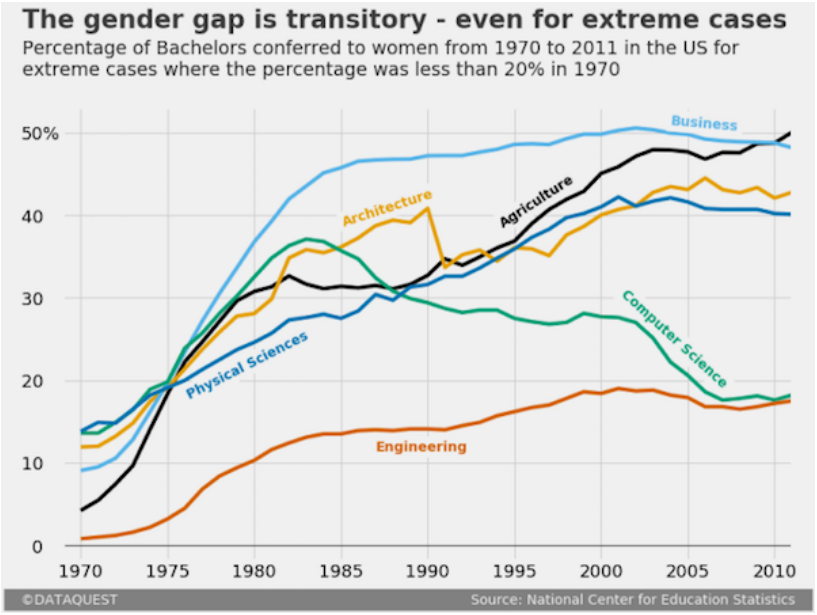
طرق أخرى في تحويل البيانات تشمل التحويلات متعدد الحدود وتحويل بوكس كوكس. 📝, 📝

مبادئ النصوص

من المهم إضافة أكبر قدر ممكن من النصوص التوضيحية لأي رسم بياني تنوي مشاركته بشكل واسع. مثلاً، الرسم البياني التالي تبدو البيانات فيه واضحة ولكن يقدم القليل من المعلومات التي تفيدنا في فهم ما تم رسمه:



لتوضيح محتوى الرسم، أضفنا عنوان، عنوان فرعي، عناوين للمحاور ووحداتها، ومسميات للخطوط التي تم رسمها:



(التدوينة هنا توضح كيف تمت إضافة كل هذه التعديلات باستخدام `matplotlib`.)

بشكل عام، نضيف التالي إذا أردنا نصوصاً توضيحية في رسمنا البياني:

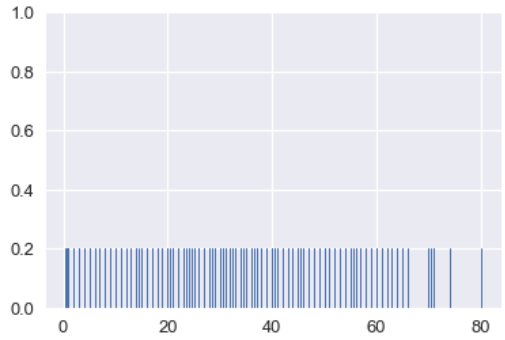
- عنوان الرسم.
- عناوين المحاور.
- إشارات للخطوط والعلامات ذات القيم المهمة.
- مسميات للنقاط المهمة والغير متوقعة.
- عنوان فرعي البيانات والمعلومات المهمة التي تقدمها.

مبادئ التبسيط

التبسيط يساعدنا في رسم البيانات بشكل واضح عندما تكون النقاط فيها كثيره. في الحقيقة سبق أن شاهدنا ذلك: المدرج التكراري هو نوع من التبسيط للرسم التخطيطي Rug Plots. الرسم التخطيطي التالي يوضح عمر كل شخص في تايتانيك:

```
ages = ti['age'].dropna()
sns.rugplot(ages, height=0.2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a20c05b38>
```

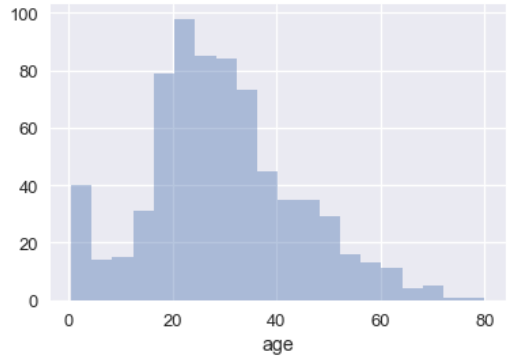


توجد الكثير من الخطوط التي تصعب علينا معرفة أين تقع أكثر البيانات. إضافة إلى انه بعض النقاط تتداخل في بعضها، ويصعب علينا ذلك معرفة كم نقطة تتجمع في 0. هذه المشكلة يطلق عليها فرط الرسم *Overplotting* ونحاول دائماً الابتعاد عن ذلك قدر الإمكان.

لإظهار توزيع البيانات، يمكننا التبديل بين إظهارها كمجموعة واحدة بشريط يمتد للأعلى عندما تكون هناك بيانات اضافيه في نفس النقطة. التبسيط يعني الخطوات لتبديل بعض النقاط بعلامات؛ نحاول عدم إظهار كل نقطه في البيانات لمحاولة الاتجاه العام للبيانات:

```
sns.distplot(ages, kde=False)
```

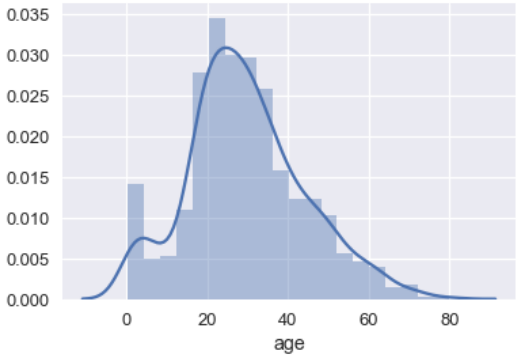
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a23c384e0>
```



رأينا أَيْضًا أن seaborn سترسم خط منحني بشكل تلقائي في المدرج التكراري:

```
sns.distplot(ages)
```

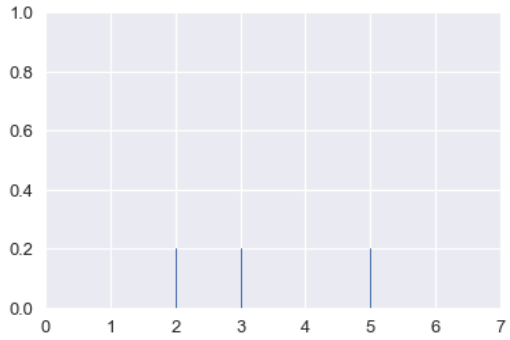
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a23d89780>
```



هذه طريقة أخرى من التبسيط تسمى تقدير كثافة النواة KDE (*Kernel Density Estimation*). بدلاً من تجميع النقاط معناً ورسم شريط، KDE تقوم برسم منحنى فوق كل نقطه وتجميعها كل منحنى وتحديد التقدير النهائي لتوزيع البيانات. لنأخذ الرسمة التخطيطية التالية لثلاث نقاط:

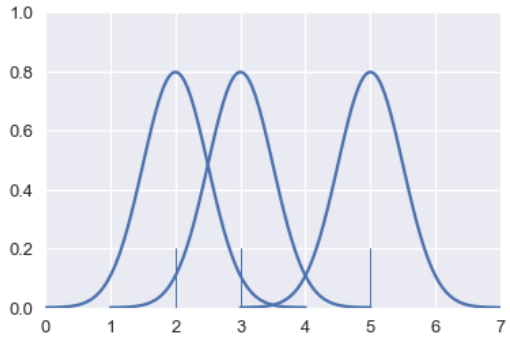
```
points = np.array([2, 3, 5])
sns.rugplot(points, height=0.2)
```

```
plt.xlim(0, 7);
```



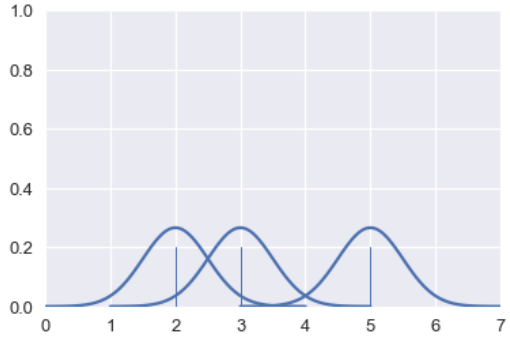
لتطبيق تقدير كثافة النقاط نقوم برسم توزيع احتمالي طبيعي Normal Gaussian في كل نقطة:

```
from scipy.stats import norm

def gaussians(points, scale=True, sd=0.5):
    x_vals = [np.linspace(point - 2, point + 2, 100) for point in points]
    y_vals = [norm.pdf(xs, loc=point, scale=sd) for xs, point in zip(x_vals, points)]
    if scale:
        y_vals = [ys / len(points) for ys in y_vals]
    return zip(x_vals, y_vals)

for xs, ys in gaussians(points, scale=False):
    plt.plot(xs, ys, c=sns.color_palette()[0])

sns.rugplot(points, height=0.2)
plt.xlim(0, 7)
plt.ylim(0, 1);
```



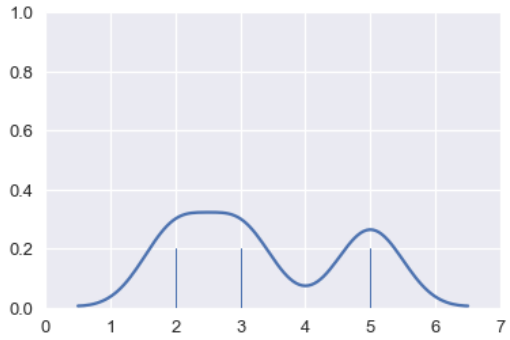
المساحة أسفل المنحنى الاحتمالي الطبيعي تساوي 1. بما أننا سنجمع أكثر من منحنى معاً، سنقوم بإعادة حساب مساحة كل منحنى حتى يكون مجموع كلا المنحنيات يساوي 1.

```
for xs, ys in gaussians(points):
    plt.plot(xs, ys, c=sns.color_palette()[0])

sns.rugplot(points, height=0.2)
plt.xlim(0, 7)
plt.ylim(0, 1);
```



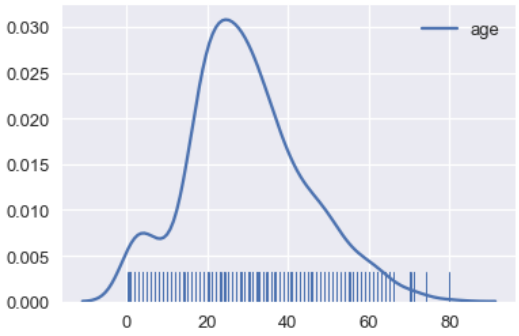
أَخِيرًا، نجمع كل المنحنيات معاً إيجاد المنحنى الأخير للتوزيع:

```
sns.rugplot(points, height=0.2)
sns.kdeplot(points, bw=0.5)
plt.xlim(0, 7)
plt.ylim(0, 1);
```



بإتباع نفس الطريقة، يمكننا استخدام KDE لتبسيط الكثير من النقاط:

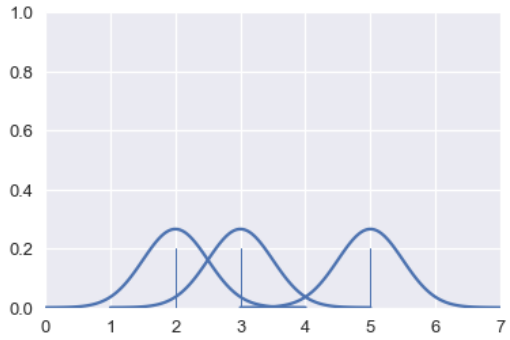
```
# أظهار النقاط الاصلية قبل التبسيط
sns.rugplot(ages, height=0.1)

# اظهار تقدير التبسيط للتوزيع
sns.kdeplot(ages);
```



تفاصيل KDE

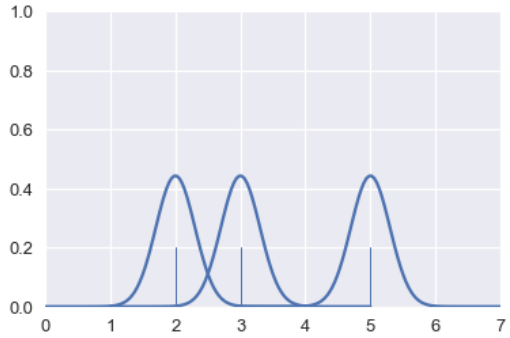
في المثال السابق، قمنا برسم منحنى بسيط فوق كل نقطة وجمعناهم معاً:

```
for xs, ys in gaussians(points):
    plt.plot(xs, ys, c=sns.color_palette()[0])

sns.rugplot(points, height=0.2)
plt.xlim(0, 7)
plt.ylim(0, 1);
```



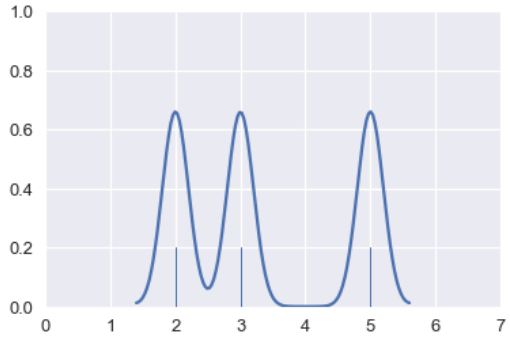
يمكننا التحكم بعرض المنحنيات. مثلاً، يمكننا تصغير حجمها. يسمى ذاك تقليل حجم نطاق *Bandwidth* تقدير الكثافة.

```
for xs, ys in gaussians(points, sd=0.3):
    plt.plot(xs, ys, c=sns.color_palette()[0])

sns.rugplot(points, height=0.2)
plt.xlim(0, 7)
plt.ylim(0, 1);
```

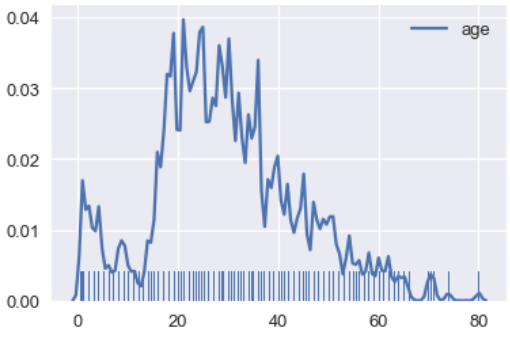


عندما نجمع المنحنيات ذات العرض الأقل معاً، نُنتِج تقدير نهائي أكثر تفصيلاً:

```
sns.rugplot(points, height=0.2)
sns.kdeplot(points, bw=0.2)
plt.xlim(0, 7)
plt.ylim(0, 1);
```



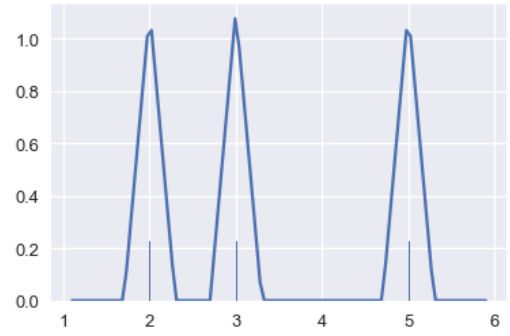

```
# رسم KDE اقل نطاق باستخدام تايتانيك ركاب لأعمار
sns.rugplot(ages, height=0.1)
sns.kdeplot(ages, bw=0.5);
```



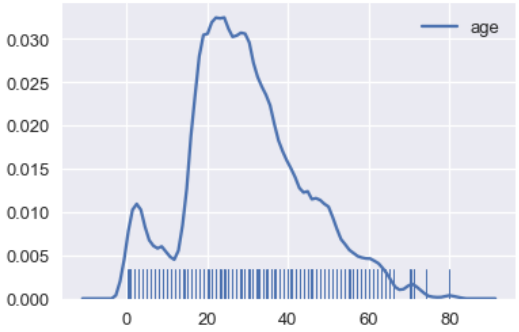
تماماً مثل التعديل في الأشرطة في المخطط الشريطي، عادة ما نقوم بالتعديل على النطاق حتى نرى أن الرسم البياني النهائي يظهر التوزيع دون تشتيت انتباه المتلقي بكثير من التفاصيل.

على الرغم من رسمنا لمنحنى في كل نقطة، يمكننا استخدام دوال أخرى للتقدير. يسمى ذلك تغير *نواة Kernal* تقدير كثافة النواة. سابقاً، استخدمنا نواة غاوسية *Gaussian Kernal*. الآن، سنستخدم نواة ثلاثية النقاط *Triangular Kernal* التي ترسم خطوط منحدره قبل وبعد كل نقطة.

```
sns.rugplot(points, height=0.2)
sns.kdeplot(points, kernel='tri', bw=0.3)
```




```
# رسم KDE لأعمار ركاب تايتانيك باستخدام نواه ثلاثية النطاق
sns.rugplot(ages, height=0.1)
sns.kdeplot(ages, kernel='tri');
```



تفاصيل تبسيط مخططات التشتت

يمكننا تبسيط الرسوم ثنائية الأبعاد عندما نواجه مشكلة فرط الرسم.

المثال التالي تأتي بياناته من سباق أزهار الكرز، سباق سنوي ل 10 أميال في مدينة واشنطن العاصمة. كل متسابق سُجل عمره ووقت نهايته للسباق؛ قمنا برسم كل البيانات في مخطط التشتت التالي:

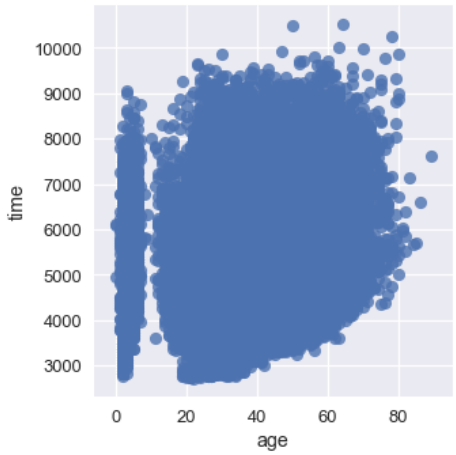
لتحميل بيانات سباق أزهار الكرز اضغط هنا.

```
runners = pd.read_csv('data/cherryBlossomMen.csv').dropna()
runners
```

	year	place	age	time
0	1999	1	28	2819
1	1999	2	24	2821
2	1999	3	27	2823
...	...	...	...	...
70066	2012	7190	56	8840
70067	2012	7191	35	8850
70069	2012	7193	48	9059

70045 rows × 4 columns

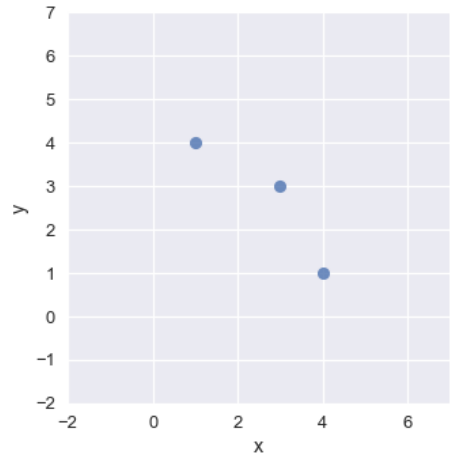
```
sns.lmplot(x='age', y='time', data=runners, fit_reg=False);
```



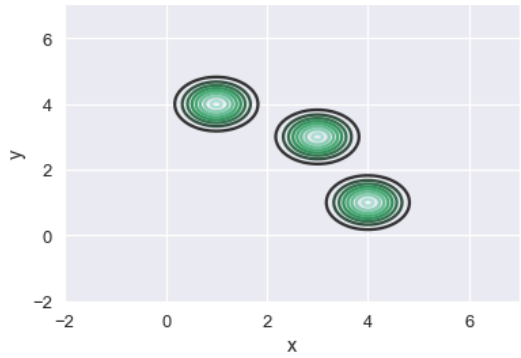
الكثير من النقاط متداخلة في بعضها البعض ويصعب بهذا الشكل تحديد اتجاه البيانات!

يمكننا تبسيط مخطط التشتت باستخدام KDE للرسوم ثنائية الأبعاد. عندما نطبقها على الرسمه، نقوم برسم منحنى ثلاثي الأبعاد على كل نقطه. بهذه الطريقة سيكون شكل المنحنى كالجبل ومتجه إلى الخارج:

```
# رسم ثلاث نقاط
two_d_points = pd.DataFrame({'x': [1, 3, 4], 'y': [4, 3, 1]})
sns.lmplot(x='x', y='y', data=two_d_points, fit_reg=False)
plt.xlim(-2, 7)
plt.ylim(-2, 7);
```



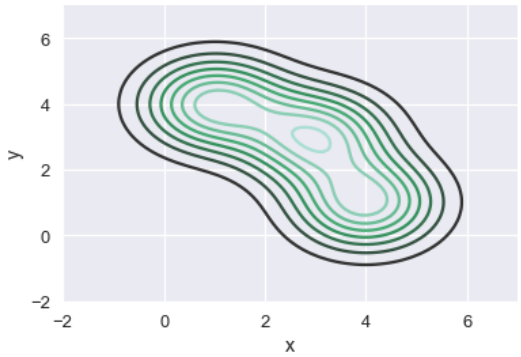

```
# رسم المنحنى لكل نقطه واستخدام مخطط المنسوب لإظهار كل واحد
sns.kdeplot(two_d_points['x'], two_d_points['y'], bw=0.4)
plt.xlim(-2, 7)
plt.ylim(-2, 7);
```



كما رأينا سابقاً، نعيد حساب المساحة لكل منحنى ونجمعها معاً حتى نصل لمخطط منسوبي *Contour Plot* نهائي لمخطط التشتت:

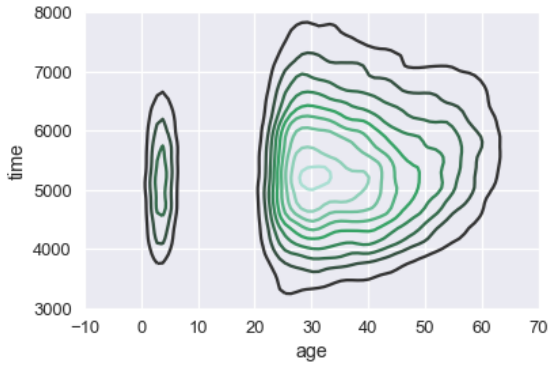
```
```

```
sns.kdeplot(two_d_points['x'], two_d_points['y'])
plt.xlim(-2, 7)
plt.ylim(-2, 7);
```



الرسمه النهائية توضح الميلان نحو الأسفل لكل الثلاث نقاط. بنفس الطريقة، يمكننا تطبيق KDE لتبسيط مخطط التشتت على بيانات الأعمار ووقت انتهاء المتسابقين للسباق:

```
sns.kdeplot(runners['age'], runners['time'])
plt.xlim(-10, 70)
plt.ylim(3000, 8000);
```



يمكن أن نلاحظ أن أكثر المتسابقين تتراوح أعمارهم بين 25 و 50 سنه، وأكثر المتسابقين استغرقوه 400 إلى 700 ثانيه (بين ساعة وساعتين) حتى ينهون السباق.

نلاحظ أَيْضًا وبشكل واضح أن هناك مجموعه من المتسابقين مشكوك بأمرها الذين بين 0 و 10 سنوات. ربما نحتاج إلى التحقق بشكل أكثر من بيانات الأعمار مما إذا كانت مسجله بشكل صحيح.

يمكن أَيْضًا أن نلاحظ بأن الوقت لإنهاء السباق يتجه للأعلى كلما زاد عمر المتسابق

فلسفة تصوير البيانات

للأسف، الكثير من الرسومات البيانية حول العالم لا تطبق مبادئنا في تصوير البيانات. هذه صفحة لرسومات بيانيه تظهر على الصفحة الأولى من بحث قوقل ل "good data visualization" تم أخذ الصورة في ربيع 2018. كم من الرسوم البيانية السيئة تلاحظ؟

الرسم البياني الفعال يظهر البيانات بشكل أوضح. كل مبادئنا لرسم البيانات تهدف لجعل البيانات مفهومه للقارئ. تستخدم الرسوم البيانية أَحيَانًا للتضليل وإظهار معلومات خاطئة. عندما يتم رسمها بشكل صحيح، الرسوم البيانية واحدة من أهم الأدوات لاكتشاف، توضيح والربط بين اتجاهات البيانات والشذوذ فيها.

تقنيات الويب

مقدمة

قبل ظهور الإنترنت، كان علماء البيانات يقومون بنقل الأقراص الصلبة بينهم يدوياً لمشاركة البيانات. الآن، يمكننا بسهوله البحث واستيراد البيانات من أجهزة الكمبيوتر حول العالم.

على الرغم من أننا نستخدم الإنترنت لتحميل ومشاركة ملفات البيانات، مواقع الإنترنت أيضا تحتوي على عدد كبير من المعلومات محفوظة كنصوص، صور أو فيديو. بتعلم تقنيات الويب، يمكننا أن نستخدمها كمصدر للبيانات. في هذا الفصل، سنتعرف على HTTP، الطريقة الأساسية للتواصل في الويب، و XML/HTML، التنسيقات الأساسية للملفات في مواقع الإنترنت.

HTTP

ال HTTP (أو **H**yper**T**ext **T**ransfer **P**rotocol) هي وسيلة وبروتوكول للطلب والرد وتسمح لجهاز كمبيوتر بالتواصل مع جهاز آخر عبر الإنترنت.

الطلب والرد

يسمح الإنترنت لأجهزة الكمبيوتر بإرسال النصوص بينها، ولكن دون أي قيود عن نوع النصوص. تقوم HTTP بإنشاء هيكل للنصوص أثناء التواصل بين الجهاز الأول (العميل) والجهاز الثاني (الخادم). في هذا البروتوكول، العميل يقدم طلب *Request* إلى الخادم، مصمم بشكل نصي محدد. يقوم الخادم بالرد *Response* بنص إلى العميل.

أداة سطر الأوامر curl تقدم لنا وسيلة سهله لإرسال طلبات HTTP. في النتيجة التالية، السطر الذي يبدأ ب < يعني أن النص أرسل في الطلب؛ الأسطر الباقية هي رد الخادم:

```
$ curl -v https://httpbin.org/html
```

كما ذكرت مُسبقاً في الفصل الثالث، لتشغيل أوامر sh على أجهزة ويندوز قم بتحميل Gitbash

```
> GET /html HTTP/1.1
> Host: httpbin.org
> User-Agent: curl/7.55.1
```

```
> Accept: */*
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< Server: meinheld/0.6.1
< Date: Wed, 11 Apr 2018 18:15:03 GMT
<
<html>
  <body>
    <h1>Herman Melville - Moby-Dick</h1>
    <p>
      Availing himself of the mild...
    </p>
  </body>
</html>
```

تشغيل الأمر `curl` يجعل جهاز العميل يكتب رسالة تبدو بهذا الشكل:

```
GET /html HTTP/1.1
Host: httpbin.org
User-Agent: curl/7.55.1
Accept: */*
{blank_line}
```

تتبع هذه الرسالة شكلاً معيناً: تبدأ ب `GET /html HTTP/1.1` والتي تعني أن الرسالة من نوع طلب HTTP `GET` إلى صفحة `/html`. الأسطر الثلاثة التي تلي سطر رأس HTTP، هي معلومات اختيارية ترسلها `curl` إلى الخادم. رؤوس HTTP Headers شكلها كالتالي `{name}: {value}`. أخيرا، السطر الفارغ في نهاية الرسالة يخبر الخادم أن الرسالة انتهت بعد ثلاث رسائل. لاحظ أننا أشرنا للسطر الفارغ ب `{blank_line}` في المثال؛ ولكن في رسالة حقيقية تستبدل `{blan_line}` بسطر فارغ دون أي نص.

إذا، جهاز العميل يستخدم الإنترنت لإرسال رسالة إلى خادم الويب `https://httpbin.org`. الخادم يعالج الطلب، ثم يقوم بإرسال الرد التالي:

```
HTTP/1.1 200 OK
Connection: keep-alive
Server: meinheld/0.6.1
Date: Wed, 11 Apr 2018 18:15:03 GMT
{blank_line}
```

السطر الأول من الرد يشير على أن الطلب تم بنجاح. الأسطر الثلاثة التي تلي سطر HTTP، هي معلومات اختيارية يرسلها الخادم إلى العميل. أخيرا السطر الفارغ يخبر جهاز العميل أن الخادم أنهى رسالته الأولية ثم سيقوم بإرسال المحتوى Body:

```
<html>
  <body>
    <h1>Herman Melville - Moby-Dick</h1>
    <p>
      Availing himself of the mild...
    </p>
  </body>
</html>
```

بروتوكول HTTP يستخدم بواسطة أغلب التطبيقات التي تتعامل مع الإنترنت. مثلاً، زيارة الموقع https://httpbin.org/html عبر متصفح يرسل طلب HTTP مشابه لطلب `curl`. بدلاً من إظهار الرد كنص فقط كما شاهدنا مسبقاً، يتعرف المتصفح أن النص عبارة عن ملف HTML فيظهره بالشكل المطلوب.

عملياً، لن نكتب طلب HTTP كامل كنص. بدلاً من ذلك، سنستخدم أدوات مثل `curl` أو مكتبات في بايثون لبناء الطلب عنا.

بايثون

مكتبة **Requests** في بايثون تسمح لنا بإنشاء طلب HTTP. الكود البرمجي التالي ينشأ طلب HTTP مشابه للطلب `curl -v https://httpbin.org/html`:

```
import requests

url = "https://httpbin.org/html"
response = requests.get(url)
response
```

```
<Response [200]>
```

الطلب

لنلقي نظره أقرب على الطلب السابق. يمكننا تصفح الطلب الرئيسي باستخدام كائن `response`؛ سنظهر رأس طلب HTTP:

```
request = response.request
for key in request.headers: # رأس HTTP في الرد محفوظ في مصفوفه
    print(f'{key}: {request.headers[key]}')
```

```
User-Agent: python-requests/2.12.4
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
```

كل طلب عبر HTTP لديه نوع. في الحالة السابقة، استخدمنا طلب GET والذي يجلب بيانات من الخادم.

```
request.method
```

```
`GET`
```

الرد

لنتحقق من الرد الذي حصلنا عليه من الخادم. أولاً، سنطبع رأس HTTP للرد:

```
for key in response.headers:
    print(f'{key}: {response.headers[key]}')
```

```
Connection: keep-alive
Server: gunicorn/19.7.1
Date: Wed, 25 Apr 2018 18:32:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 3741
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
X-Powered-By: Flask
X-Processed-Time: 0
Via: 1.1 vegur
```

يحتوي رد HTTP على رمز للحالة، رقم خاص يبين إذا نجح أو فشل الطلب. الرمز 200 يعني أن الطلب تم بنجاح:

```
response.status_code
```

```
200
```

أخيرا، سنظهر أول 100 حرف من محتوى الرد (الرد الكامل محتواه كبير ولا يظهر بشكل جيد هنا):

```
response.text[:100]
```

```
'<!DOCTYPE html>\n<html>\n  <head>\n  </head>\n  <body>\n      <h1>Herman Melville - Moby-Dick</h1>\n\n
```

أنواع الطلبات

الطلب السابق كان طلب GET. هناك عدة أنواع من طلبات HTTP؛ أكثر الطلبات أهمية هي GET و POST.

طلب GET

طلب GET يستخدم لاستيراد معلومات من الخادم. بما أن المتصفح يقوم بطلب GET كل مرة تدخل فيها رابط لموقع، يعتبر لذلك طلب GET من أكثر الطلبات استخداماً من طلبات HTTP.

تستخدم curl طلب GET دائماً، الأمر curl https://www.google.com يقوم بإنشاء طلب GET ل https://www.google.com.

طلب POST

الطلب POST يستخدم لإرسال معلومات من العميل إلى الخادم. مثلاً، بعض المواقع تحتوي على حقول نحتاج أن نملأها، مثل حقول تسجيل الدخول. عند الضغط على زر إدخال، أكثر المتصفحات ستقوم بإنشاء طلب POST لإرسال البيانات الحقول إلى الخادم ليجري العمليات عليها.

لنرى مثلاً على طلب POST يرسل الكلمة sam في المتغير name. يمكن فعل ذلك كالتالي curl -d 'name=sam' https://httpbin.org/post في سطر الأوامر.

لاحظ أن طلبنا لديه محتوى الآن (عندما عبئنا المتغير في طلب POST)، ومحتوى الرد سيكون مختلف عن محتوى رد GET السابق.

كما في رؤوس HTTP، تستخدم طلبات POST نفس الشكل مفتاح-قيمه key-value. في بايثون، نستخدم requests.post مع تمرير المتغيرات كمصفوفات لإنشاء طلبات POST:

```
post_response = requests.post("https://httpbin.org/post",
                              data={'name': 'sam'})
post_response
```

```
<Response [200]>
```

الخادم سيرد برمز للحالة لينوه إذا ما كانت عملية طلب POST تمت بنجاح. أيضا، يرسل الخادم عادة محتوى ليعرض للعميل:

```
post_response.status_code
```

```
200
```

```
post_response.text
```

```
'{\n  "args": {}, \n  "data": "", \n  "files": {}, \n  "form": {\n    "name": "sam"\n  }, \n  "headers": {
```

أنواع رموز الحالة للردود

ردود HTTP السابقة تحتوي على رمز 200 لحالة الرد. هذا الرمز يخبرنا أن الطلب تم بنجاح. هناك الكثير من الرموز الأخرى لحالات الرد ل HTTP. لحسن حظنا أن تم جمعها في مجموعات لتكون أسهل لنا للحفظ: 📝

- ردود 100: معلوماتية: رد على أنه مطلوب معلومات أكثر من الخادم أو العميل. (مثلاً 100 استمرار، 102 جاري العمل).
- ردود 200: نجاح: طلب العميل تم بنجاح. (مثلاً 200 نجاح، 202 مقبول).
- ردود 300: إعادة توجيه: الرابط URL المطلوب موجود بمكان آخر؛ ربما يحتاج لقرار إضافي من المستخدم. (مثلاً 300 اختيارات، 301 منقول بشكل كامل).
- ردود 400: خطأ من العميل: خطأ من جهة العميل (مثلاً 400 طلب خاطئ, 403 محظور, 404 غير موجود).
- ردود 500: خطأ من الخادم: خطأ من جهة الخادم أو أن الخادم لا يمكنه من إجراء الطلب (مثلاً 500 خطأ داخلي في الخادم، 503 الخدمة غير موجودة)

يمكننا عرض بعض الأمثلة على الأخطاء:

```
# 404 هذه الصفحه غير موجوده، سنحصل على رمز الخطأ بعدم وجود الصفحه
url = "https://www.youtube.com/404errorwow"
errorResponse = requests.get(url)
print(errorResponse)
```

```
<Response [404]>
```

```
# رد الرد لهذه الصفحه 500 خطأ في الخادم
url = "https://httpstat.us/500"
```

```
serverResponse = requests.get(url)
print(serverResponse)
```

```
<Response [500]>
```

ملخص

تعرفنا على بروتوكولات HTTP، الطرق البسيطة للتواصل بين البرامج التي تستخدم الويب. على الرغم أن البروتوكول يحدد شكل معين للنصوص، ننتقل عادةً لأدوات أخرى لكتابة طلبات HTTP لنا، مثل أداة سطر الأوامر `curl` أو مكتبة `requests` في بايثون.

التعامل مع النصوص

مقدمة

الكثير من البيانات التي نواجهها في ملف CSV ليست أرقام بس نصوص موجودة في كُتب، ملفات، تدوينات، أو تعليقات في الإنترنت. كما في الكثير من أنواع البيانات، هناك طرق مختلفة للتعامل مع النصوص وقد نحتاج لكتابة أكثر من كتاب لشرحها بشكل تفصيلي. في هذا الفصل، سنشرح جزء بسيط من هذه التقنيات والتي توفر لنا العديد من العمليات المفيدة للعمل مع النصوص: معالجة نصوص بايثون والتعابير النمطية RegEx.

دوال النصوص في بايثون

توفر لنا بايثون العديد من الدوال للتعامل مع النصوص. رغم بساطتها، تعد هذه الدوال من الأولويات في التعامل مع النصوص والتي جُمعت مع بعضها البعض لتكون دوال أكثر تعقيداً. سنقوم بشرح دوال بايثون للتعامل مع النصوص حسب كثرة الاستخدام في تنظيف البيانات النصية.

تنظيف البيانات النصية

تأتي البيانات عادةً من عدة مصادر وكل مصدر لديه طريقه مختلفة لترميز المعلومات. في المثال التالي، لدينا جدول يحتوي على الولايات الأمريكية والمقاطعات التي تشملها وجدول آخر يحتوي على عدد سكان المقاطعة.

```
state = pd.DataFrame({
    'County': [
        'De Witt County',
        'Lac qui Parle County',
        'Lewis and Clark County',
        'St John the Baptist Parish',
    ],
    'State': [
        'IL',
        'MN',
        'MT',
        'LA',
    ]
})
population = pd.DataFrame({
    'County': [
        'DeWitt  ',
        'Lac Qui Parle',
        'Lewis & Clark',
        'St. John the Baptist',
    ],
    'Population': [
        '16,798',
        '8,067',
        '55,716',
        '43,044',
    ]
})
```

	County	State
0	De Witt County	IL
1	Lac qui Parle County	MN
2	Lewis and Clark County	MT
3	St John the Baptist Parish	LA

	County	Population
0	DeWitt	16,798
1	Lac Qui Parle	8,067
2	Lewis & Clark	55,716
3	St. John the Baptist	43,044

بالطبع نريد جمع جدولي `state` و `population` معاً باستخدام العمود `County`. للأسف، لا تتطابق أي من أسماء المقاطعات في الجدولين. هذا المثال يوضح عدد من المشاكل في البيانات النصية وهي كالتالي:

- الكتابة بحروف كبيرة / صغيرة: `qui` و `Qui`
- علامات ترقيم مختلفة: `St.` و `St`
- عدم كتابة بعض الكلمات: مثلاً كلمتي `County/Parish` لم يتم كتابتها في جدول `population`.
- إضافة مسافات: `DeWitt` و `De Witt`.
- استخدام مُختلف للاختصارات: `&` و `and`.

دوال النصوص

دوال النصوص في بايثون تساعدنا بالبدء في حل هذه المشاكل. هذه الدوال مُعرّفه في جميع نسخ بايثون لذا لا نحتاج لإستخادم اي مكتبه أخرى. على الرغم أنه مهم جداً لك شخصياً معرفة جميع الدوال للتعامل مع النصوص، قمنا بشرح بعض من أكثر الدوال استخداماً في الجدول التالي:

الدالة	الوصف
[str[x:y	تقوم الداله بفصل النص `str` من `x` (تشمل الحرف) حتى `y` (لا تشمل الحرف).
str.lower()	تنشئ نسخه من النص `str` بعد تحويل جميع الحروف فيه إلى حروف صغيره.
(str.replace(a, b	تحويل جميع مرات ظهور النص/الحرف `a` في `str` إلى النص/الحرف `b`.
(str.split(a	تقوم الدالة بفصل النص `str` عند ظهور النص/الحرف `a`.
str.strip()	تحذف الداله المسافات الفارغه في بداية ونهاية النص `str`

قمنا باختيار النص `St. John the Baptist` من جدولي `state` و `population` لتطبيق دوال النصوص بحذف الحروف الكبيرة، علامات الترقيم، والتخلص من الكلمتين `county` و `parish`:

```
john1 = state.loc[3, 'County']
john2 = population.loc[3, 'County']

(john1
 .lower()
 .strip()
 .replace(' parish', '')
 .replace(' county', '')
 .replace('&', 'and')
 .replace('.', '')
 .replace(' ', '')
)
```

```
'stjohnthebaptist'
```

تطبيق نفس الدوال على `john2` يؤكد لنا أن النصين مُتطابقان:

```
(john2
 .lower()
 .strip()
 .replace(' parish', '')
 .replace(' county', '')
 .replace('&', 'and')
 .replace('.', '')
 .replace(' ', '')
)
```

```
'stjohnthebaptist'
```

عند الوصول لنتيجة مُقنعه، نقوم بتعريف دالة باسم `clean_county` والتي ستطبق مهام تنظيف المدخلات (المقاطعات):

```
def clean_county(county):
    return (county
            .lower()
            .strip()
            .replace(' county', '')
```

```
            .replace(' parish', '')
            .replace('&', 'and')
            .replace(' ', '')
            .replace('.', ''))
```

يمكننا التأكد أن الدالة clean_county تقوم بتوحيد جميع أسماء المقاطعات عبر تطبيقها على الجدولين:

```
([clean_county(county) for county in state['County']],
 [clean_county(county) for county in population['County']]
)
```

```
(['dewitt', 'lacquiparle', 'lewisandclark', 'stjohnthebaptist'],
 ['dewitt', 'lacquiparle', 'lewisandclark', 'stjohnthebaptist'])
```

بما أن كلا العامودين الآن يحتويان على نفس اسم المقاطعة وبنفس الشكل، يمكننا جمع الجدولين معاً باستخدام اسم المقاطعة.

دوال النصوص في باندا

في الكود البرمجي السابق، استخدمنا التكرار Loop لتغير أسماء المقاطعات. مصفوفات pandas تقدم طريقة أسهل لتطبيق دوال النصوص لجميع محتوى المصفوفة. أولا، لنرى أسماء المقاطعات في جدول state:

```
state['County']
```

```
0               De Witt County
1         Lac qui Parle County
2        Lewis and Clark County
3    St John the Baptist Parish
Name: County, dtype: object
```

الدالة .str في مصفوفات باندا تتعامل مع النص كما في بايثون. استخدام الدوال على .str يطبقها على جميع القيم في المصفوفة:

```
state['County'].str.lower()
```

```
0               de witt county
1         lac qui parle county
2        lewis and clark county
3    st john the baptist parish
Name: County, dtype: object
```

يُمكنا ذلك من تحويل جميع النصوص في المصفوفة دون الحاجة لاستخدام التكرار:

```
(state['County']
 .str.lower()
 .str.strip()
 .str.replace(' parish', '')
 .str.replace(' county', '')
 .str.replace('&', 'and')
 .str.replace('.', '')
 .str.replace(' ', '')
)
```

```
0             dewitt
1        lacquiparle
2       lewisandclark
3    stjohnthebaptist
Name: County, dtype: object
```

نعيد حفظ نتيجة تغير شكل عمود المقاطعات على نفس العمود:

```
state['County'] = (state['County']
 .str.lower()
```

```
 .str.strip()
 .str.replace(' parish', '')
 .str.replace(' county', '')
 .str.replace('&', 'and')
 .str.replace('.', '')
 .str.replace(' ', '')
)

population['County'] = (population['County']
 .str.lower()
 .str.strip()
 .str.replace(' parish', '')
 .str.replace(' county', '')
 .str.replace('&', 'and')
 .str.replace('.', '')
 .str.replace(' ', '')
)
```

والآن بما أن كلا الجدولين لديهما نفس التعبير النصي للمقاطعات:

```
state
```

	County	State
0	dewitt	IL
1	lacquiparle	MN
2	lewisandclark	MT
3	stjohnthebaptist	LA

```
population
```

	County	Population
0	dewitt	16,798
1	lacquiparle	8,067
2	lewisandclark	55,716
3	stjohnthebaptist	43,044

من السهل جمع الجدولين عندما يتطابق عمود المناطق:

```
state.merge(population, on='County')
```

	County	State	Population
0	dewitt	IL	16,798
1	lacquiparle	MN	8,067
2	lewisandclark	MT	55,716
3	stjohnthebaptist	LA	43,044

ملخص دوال النصوص

توفر بايثون دوال سهله وعملية للتعامل وتعديل النصوص. مصفوفات بانداز توفر نفس الدوال وتسهل تطبيقها على جميع القيم داخل المصفوفة.

يمكنك تصفح شرح كامل عن دوال النصوص في بايثون هنا ودوال النصوص في بانداز هنا

التعابير النمطية RegEx

في هذا الجزء سنتحدث عن التعابير النمطية RegEx، أداة مهمة للتحقق من الأنماط في النصوص.

في النصوص الكبيرة، الكثير من النصوص الفرعية تأتي بعدة أشكال. مثلاً، الجملة في الأسفل تحتوي على رقم هاتف:

```
"give me a call, my number is 123-456-7890."
```

رقم الهاتف يحتوي على الأنماط التالية:

- ثلاث أرقام.
- متبوعه بخط فاصل.
- متبوعه بثلاث أرقام.
- متبوعه بخط فاصل.
- متبوعه بأربع أرقام.

إذا أعطينا نصاً مكتوب، قد نرغب بإيجاد وسحب أرقام الهواتف فقط. رُبَّمَا أيضا نريد جزء معين من رقم الهاتف، مثلاً، سحب رمز المنطقة (الثلاث أرقام الأولى) قد يوصل لنا مكان تواجد صاحب الرقم المذكور في النص.

للتحقق إذا كان النص يحتوي على رقم هاتف، قد نعرف دالة كالتالي:

```
def is_phone_number(string):

    digits = '0123456789'

    def is_not_digit(token):
        return token not in digits

    # ثلاث ارقام
    for i in range(3):
        if is_not_digit(string[i]):
            return False

    # متبوعه بخط فاصل
    if string[3] != '-':
        return False

    # متبوعه بثلاث ارقام
    for i in range(4, 7):
        if is_not_digit(string[i]):
            return False

    # متبوعه بخط فاصل
    if string[7] != '-':
        return False

    # متبوعه بأربع ارقام
    for i in range(8, 12):
        if is_not_digit(string[i]):
            return False

    return True
```

وللتحقق:

```
is_phone_number("382-384-3840")
```

```
True
```

```
is_phone_number("phone number")
```

```
False
```

الكود البرمجي السابق يبدو طويل وغير مرغوب فيه. بدلاً من الدوران على كل الحروف في النص، نفضل أن نحدد النمط وأوامر لبايثون تسهل علينا إيجاد الأنماط المطابقة لطلباتنا.

التعابير النمطية Regular expressions (وبشكل مختصر **RegEx**) تحل هذه المشكلة بجعلنا ننشأ الأنماط للنصوص. باستخدامها، يمكننا إعادة تعريف الدالة السابقة `is_phone_number` بسطرين في بايثون:

```
import re

def is_phone_number(string):
    regex = r"[0-9]{3}-[0-9]{3}-[0-9]{4}"
    return re.search(regex, string) is not None

is_phone_number("382-384-3840")
```

```
True
```

في المثال السابق، استخدمنا التعبير النمطي [0-9]{3}-[0-9]{3}-[0-9]{4} لمطابقة أرقام الهواتف. ربما يبدو مبهماً في البداية، ولكن أوامر التعابير النمطية سهلة للتعلم؛ في هذا الجزء شرحنا كل الأوامر.

سنقوم بالتعرف على المكتبه التي تأتي مع بايثون بأسم re الخاصه بكتابة وتطبيق عمليات النصوص بإستخدام التعابير النمطية.

كتابة أوامر التعابير النمطية

سنبدأ أولاً بالتعرف على طريقة الكتابه. التعابير النمطية عادةً ما تُحفظ على شكل نصوص خام Raw String. تعمل تماماً كما تعمل النصوص في بايثون ولكن دون قوانين خاصة للخطوط المائله \.

الخط المائل \ يستخدم لتجاهل ما بعدهEscape. مثلاً، اذا اردنا كتابة It's raining، نحتاج لإضافة \ قبل ' لأن علامة الإقتباس تستخدم لحفظ النص:

```
print('It\'s raining')
```

```
It's raining
```

تستخدم ايضاً \ لإضافة سطر جديد باستخدام \n أو اضافة مسافة باستخدام \t

مثلاً لحفظ النص hello \ world في بايثون، نحتاج لكتابة:

```
# الخطوط المائلة في بايثون تحتاج للتوضيح ليتم تجاهلها في نصوص بايثون العاديه
some_string = 'hello \\ world'
print(some_string)
```

```
hello \ world
```

استخدام النصوص الخامة يحذف شرط إضافة رمز لتجاهل الخطوط المائلة:

لإخبار بايثون أن النص التالي يعتبر نص خام Raw String, نقوم بإضافة r قبل النص

```
# لاحظ ال `r` قبل بداية النص
some_raw_string = r'hello \ world'
print(some_raw_string)
```

```
hello \ world
```

بما ان الخطوط المائلة تظهر بشكل كثير في التعابير النمطية، سنستخدم النصوص الخامة لكتابة جميع التعابير النمطية في هذا الجزء.

حرفياً

إيجاد حرف بشكل معين بالتحديد في التعبير النمطي يوجد لنا ذلك الحرف فقط. مثلاً، التعبير r"a" يوجد لنا أي "a" في Say! I like green eggs and ham!. جميع الأرقام والأحرف ورموز الترقيم يمكن البحث عنها بشكل خاص في التعابير النمطية:

```
def show_regex_match(text, regex):
    """
    تطبع النص مع تغير اللون لتحديد نمط regex
    """
    print(re.sub(f'({regex})', r'\033[1;30;43m\1\033[m', text))
```

```
regex = r"green"
show_regex_match("Say! I like green eggs and ham!", regex)
```

```
Say! I like green eggs and ham!
```

```
show_regex_match("Say! I like green eggs and ham!", r"a")
```

```
Say! I like green eggs and ham!
```

في المثال السابق، شاهدنا أن التعابير النمطية يمكنها إيجاد الأنماط أينما ظهر في النص المدخل. في بايثون، في بايثون هذه الطريقة تختلف بِنَاءًا على الدالة المدخلة لمطابقة التعبير النمطي، بعض الدوال فقط تظهر النتيجة إذا ظهر في بداية النص، والبعض يظهرها إذا ظهرت في أي مكان في النص.

لاحظ أيضا أن الدالة show_regex_match تظهر جميع تكرارات النمط في النص المدخل. مرة أخرى، يختلف ذلك حسب دالة بايثون المستخدمة، بعضها يظهر جميع التكرارات والبعض فقط يظهر أول تكرار.

التعابير النمطية تتحسس لحالة وشكل الحرف. في المثال التالي، التعبير يطابق فقط الحرف الصغير s في eggs، وليس الحرف الكبير S في Say.

```
show_regex_match("Say! I like green eggs and ham!", r"s")
```

```
Say! I like green eggs and ham!
```

رموز خاصة

بعض الرموز لديها معاني خاصة في RegEx. هذه الرموز تساعدنا على إيجاد أنماط مختلفة من النصوص.

الرمز (نقطة) تعني أنه تطابق أي حرف عدى أن يكون ببداية سطر جديد:

```
show_regex_match("Call me at 382-384-3840.", r".all")
```

```
Call me at 382-384-3840.
```

لنطابق النقطة بشكل خاص يجب علينا تجاهلها كرمز بإضافة \:

```
show_regex_match("Call me at 382-384-3840.", r"\.")
```

```
Call me at 382-384-3840.
```

باستخدام النقطة يمكننا مطابقة أنواع مختلفة من الأنماط، سنقوم بكتابة تعبير نمطي لمطابقة أرقام الهواتف. مثلاً، يمكنا أخذ الرقم 382-384-3840 وتبديل كل رقم بنقطة .، لتبقى لنا فقط الخطوط الفاصلة. النتيجة كالتالي ...-...-....:

```
show_regex_match("Call me at 382-384-3840.", "...-...-....")
```

```
Call me at 382-384-3840.
```

بما أن النقطة تطابق جميع النصوص، النص التالي سيظهر لنا نتيجة خاطئة، لأننا نريد إيجاد الأرقام فقط وليس الحروف:

```
show_regex_match("My truck is not-all-blue.", "...-...-....")
```

```
My truck is not-all-blue.
```

المجموعات النصية

تسمح لنا المجموعات النصية بمطابقة نوع معين من النصوص، يوفر لنا ذلك إمكانية إنشاء قيود أكثر للتطابق بدلاً من النقطه . لمطابقة أي نص.
لإنشاء مجموعة، قم بإضافة الحروف داخل أقواس [ ]:

```
show_regex_match("I like your gray shirt.", "gr[ae]y")
```

```
I like your gray shirt.
```

```
show_regex_match("I like your grey shirt.", "gr[ae]y")
```

```
I like your grey shirt.
```

```
# لا يطابق، المجموعات تطابق حرف واحد فقط من ما تحتويها وليس جميعها
show_regex_match("I like your graey shirt.", "gr[ae]y")
```

```
I like your graey shirt.
```

```
# في هذا المثال، كتابة المجموعه مره اخرى ستطابقها
show_regex_match("I like your graey shirt.", "gr[ae][ae]y")
```

```
I like your graey shirt.
```

في مجموعات النصوص، الرمز . سيتخدم كحرف وليس كرمز:

```
show_regex_match("I like your grey shirt.", "irt[.]")
```

```
I like your grey shirt.
```

توجد بعض الحالات المختصرة الخاصة التي يمكن استخدامها لمطابقة مجموعات النصوص:

الإختصار	المعنى
[0-9]	جميع الارقام
[a-z]	الأحرف الصغيره
[A-Z]	الأحرف الكبيره

```
show_regex_match("I like your gray shirt.", "y[a-z]y")
```

```
I like your gray shirt.
```

تُمكنا المجموعات النصية من إنشاء أنماط خاصة بأرقام الهواتف:

```
# استبدلنا النقاط `.` في ...-...-.... ب [0-9] لتحديد النمط على الأرقام
phone_regex = r'[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]'
show_regex_match("Call me at 382-384-3840.", phone_regex)
```

```
Call me at 382-384-3840.
```

```
# والآن لا نطابق هذا النص
show_regex_match("My truck is not-all-blue.", phone_regex)
```

```
My truck is not-all-blue.
```

عكس المجموعات النصية

عكس المجموعات يطابق أي نص غير الموجود في المجموعة. لإنشاء هذا التعبير، نضيف النص داخل الأقواس [^]:

```
show_regex_match("The car parked in the garage.", r"[^c]ar")
```

```
The car parked in the garage.
```

الحدود الرقمية

لإنشاء تعبير نمطي لمطابقة أرقام الهواتف، كتبنا التالي:

```
[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]
```

هذا النمط يطابق ثلاثة أرقام، متبوعه بخط، ثم ثلاثة أرقام وخط ثم أربعة أرقام أخرى.

الحدود الرقمية تسمح لنا مطابقة التكرار في النمط. نحدد عدد مرات التكرار بوضع الرقم داخل أقواس معقوفة { }:

```
phone_regex = r'[0-9]{3}-[0-9]{3}-[0-9]{4}'
show_regex_match("Call me at 382-384-3840.", phone_regex)
```

```
Call me at 382-384-3840.
```

```
# لا تطابق لأن اول جزء مكون من رقمين
phone_regex = r'[0-9]{3}-[0-9]{3}-[0-9]{4}'
show_regex_match("Call me at 12-384-3840.", phone_regex)
```

```
Call me at 12-384-3840.
```

الحدود تطابق جميع النصوص أو الفئات النصية التي كُتبت قبلها. الجدول التالي يظهر شرح لها:

رمز الحدود الرقمية	المعنى
{m, n}	مطابقة تكرار الحرف من m إلى n مرات.
{m}	مطابقة تكرار الحرف m مرات.
{,m}	مطابقة تكرار الحرف على الأقل m مرات.
{n,}	مطابقة تكرار الحرف بالأكثر n مرات.

اختصارات الحدود الرقمية

الكثير من الحدود الرقمية التي تستخدم كثيراً تم اختصارها:

الرمز	رمز الحدود الرقمية المفصل	المعنى
*	{,0}	مطابقة تكرار الحرف 0 أو أكثر.
+	{,1}	مطابقة تكرار الحرف مرة واحدة أو أكثر.
?	{0,1}	مطابقة تكرار الحرف 0 أو مرة واحدة.

نستخدم الرمز * بدلاً من {,0} في الأمثلة التالية:

```
# ثلاث مرات a تكرار
show_regex_match('He screamed "Aaaah!" as the cart took a plunge.', "Aa*h!")
```

```
He screamed "Aaaah!" as the cart took a plunge.
```

```
# مطابقة المزيد من a
show_regex_match(
    'He screamed "Aaaaaaaaaaaaaaaaaaah!" as the cart took a plunge.',
    "Aa*h!"
)
```

```
He screamed "Aaaaaaaaaaaaaaaaaaah!" as the cart took a plunge.
```

```
# عدم وجود a بشكل صغير
show_regex_match('He screamed "Ah!" as the cart took a plunge.', "Aa*h!")
```

```
He screamed "Ah!" as the cart took a plunge.
```

الحدود الكمية جشعه!

توجد لنا دائماً الحدود الكمية أكثر القيم تطابقاً. تظهر بعض المرات نتائج غير متوقعة:

```
# حاولنا إيجاد 311 و 911 ولكن تم ايجاد and ايضاً
# لأن
# <311> and <911>
# هي اعلى قيمه ممكنه للنمط <.+>
show_regex_match("Remember the numbers <311> and <911>", "<.+>")
```

```
Remember the numbers <311> and <911>
```

في هذا المثال نلاحظ أن التعبير النمطي لم يتوقف عند أول ظهور ل > بس أستمر إلى آخر تكرار، يقصد هنا أن الحدود الكمية تبحث دائماً عن المزيد ولا تتوقف عند تكرار واحد إلا إذا حددنا ذلك

في كثير من الحالات، استخدام مجموعات النصوص يسهل علينا تفادي هذه النتائج الخاطئة:

```
show_regex_match("Remember the numbers <311> and <911>", "<[0-9]+>")
```

```
Remember the numbers <311> and <911>
```

تحديد المكان

في بعض الأحيان نريد أن يتطابق النمط إذا كان في بداية أو نهاية النص. الرمز الخاص ^ يجعل التعبير يطابق النمط عندما يظهر في بداية النص فقط؛ والرمز الآخر الخاص $ يطابق النمط إذا ظهر في نهاية النص. مثلاً التعبير النمطي `well$` فقط يطابق ظهور كلمة `well` في نهاية النص:

```
show_regex_match('well, well, well', r"well$")
```

```
well, well, well
```

باستخدام كلا الرمزين ^ و $ يجعل التعبير يطابق النمط كاملاً:

```
phone_regex = r"^[0-9]{3}-[0-9]{3}-[0-9]{4}$"
show_regex_match('382-384-3840', phone_regex)
```

```
382-384-3840
```

```
# لا يطابق
# لأنه يبدأ بنص وليس ثلاثة أرقام
# وينتهي بنقطة وليس اربع ارقام
show_regex_match('You can call me at 382-384-3840.', phone_regex)
```

```
You can call me at 382-384-3840.
```

تجاهل الرموز الخاصة

كل الرموز الخاصة لها وظائف معينة في التعبير النمطي. ولمطابقة أحد هذه الرموز، يمكننا تجاهل وظيفتها في التعبير النمطي باستخدام \:

```
show_regex_match("Call me at [382-384-3840].", "\[")
```

```
Call me at [382-384-3840].
```

```
show_regex_match("Call me at [382-384-3840].", "\.")
```

```
Call me at [382-384-3840].
```

مرجع التعابير النمطية

قمنا بالحديث عن أهم النقاط والرموز في التعابير النمطية. ليكون لدينا مرجع كامل، أضفنا الجدول التالي.

مرجع الرموز الخاصة

هذا الجدول يحتوي على أهم الرموز الخاصة، والتي تساعدك على تحديد أنماط محددة تريد مطابقتها والبحث عنها في النصوص:

الرمز	الوصف	مثال التعبير النمطي	يطابق	لا يطابق
.	يطابق أي حرف عدا سطر جديد	...	abc	ab
				abcd
[ ]	أي حرف داخل الأقواس	[cb.]ar	car	jar
			ar.	
[^]	أي حرف غير ما داخل الأقواس	[^b]ar	car	bar
			par	ar
*	صفر أو أكثر تكرار لما يسبق الرمز	[pb]*ark	bbark	dark
			ark	
+	واحد أو أكثر تكرار لما يسبق الرمز	[pb]+ark	bbpark	dark
			bark	ark
?	صفر أو مرة واحد تكرار لما يسبق الرمز	s?he	she	the
			he	
{n}	مطابقة عدد مرات التكرار n	{hello{3	hellooo	hello
|	مطابقة أي نمط بين خيارين	we|[ui]s	we	e
			us	s
			is	
\	تجاهل الرموز الخاصة	\[hi\]	[hi]	hi

الرمز	الوصف	مثال التعبير النمطي	يطابق	لا يطابق
^	بداية السطر	^ark	ark two	dark
$	نهاية السطر	ark$	noahs ark	noahs arks

مرجع الاختصارات

بعض من الفئات المختصرة:

الرمز المختصر	التعبير النمطي الكامل للفئة	الوصف
\w	[a-zA-Z0-9]	أي حرف صغير/كبير أو رقم
\W	[[^a-zA-Z0-9	غير أي حرف صغير/كبير أو رقم
\d	[0-9]	ارقام
\D	[[^0-9	غير الأرقام
\s	[{[\t\n\f\r\p{Z	المسافات
\S	[{[^\t\n\f\r\p{z	غير المسافات

ملخص التعابير النمطية RegEx

في جميع لغات البرمجة توجد مكاتب خاصة لمطابقة الأنماط باستخدام التعابير النمطية، مما يجعلها مهمة مع اختلاف اللغات. في هذا الجزء، تحدثنا عن طريقة كتابة التعابير النمطية واهم الرموز المستخدمة.

التعابير النمطية في بايثون وبانداز

في هذا الجزء، سنتعرف على استخدام مكتبة re التعامل مع التعابير النمطية في بايثون. بما أننا تحدثنا على جزء بسيط من أكثر الدوال المستخدمة، يمكنك العودة دائماً إلى الصفحة الرئيسية لشرح مكتبة re.

RE.SEARCH

الدالة `re.search(pattern, string)` تطابق جميع الأنماط داخل المتغير `pattern` في النص `string`. يكون نتيجة الدالة أما النص إذا وجد تطابق أو `None` إن لم يجد:

```
phone_re = r"[0-9]{3}-[0-9]{3}-[0-9]{4}"
text  = "Call me at 382-384-3840."
match = re.search(phone_re, text)
match
```

```
<_sre.SRE_Match object; span=(11, 23), match='382-384-3840'>
```

على الرغم أن الكائن الذي أنشئ نتيجة لعملية البحث يحتوي على الكثير من الخصائص المفيدة، نحن عادةً نستخدم `re.search` للاختبار والتأكد في حال ظهور النمط في النص:

```
if re.search(phone_re, text):
    print("Found a match!")
```

```
Found a match!
```

```
if re.search(phone_re, 'Hello world'):
    print("No match; this won't print")
```

دالة أخرى كثيره الاستخدام هي `re.match(pattern, string)` وتعمل بنفس طريقة عمل `re.search` ولكن تقوم بالبحث عن تطابق فقط في بداية النص `string` وليس في أي جزء آخر منه.

RE.FINDALL

نستخدم الدالة `re.findall(pattern, string)` لإيجاد جميع النتائج التي تطابق النمط. هذه الدالة تنتج لنا مصفوفة تحتوي على جميع المطابقات للنمط الذي يكتب في المتغير `pattern` في النص الذي يكتب في المتغير `string`:

```
gmail_re = r'[a-zA-Z0-9]+@gmail\.com'
text = '''
From: email1@gmail.com
To: email2@yahoo.com and email3@gmail.com
'''
re.findall(gmail_re, text)
```

```
['email1@gmail.com', 'email3@gmail.com']
```

مجموعات RegEx

استخدام مجموعات RegEx يسمح لنا بمطابقة أنماط متعددة واستخراجها منفصله عن طريق إضافتها داخل أقواس ( ). عندما يحتوي التعبير النمطي على مجموعات، الدالة `re. findall` تنتج لنا مصفوفة Tuple تحتوي على نتيجة النمط مفصله.

مثلاً، التعبير النمطي التالي يستخرج أرقام الهواتف من النص:

```
phone_re = r"[0-9]{3}-[0-9]{3}-[0-9]{4}"
text  = "Sam's number is 382-384-3840 and Mary's is 123-456-7890."
re.findall(phone_re, text)
```

```
['382-384-3840', '123-456-7890']
```

لنقوم بفصل كل ثلاث أو أربع أرقام لوحدها من رقم الهاتف، يمكننا جمع كل مجموعة أرقام داخل أقواس:

```
نفس التعبير السابق مع اضافة اقواس بين كل 3/4 ارقام #
phone_re = r"([0-9]{3})-([0-9]{3})-([0-9]{4})"
text  = "Sam's number is 382-384-3840 and Mary's is 123-456-7890."
re.findall(phone_re, text)
```

```
[('382', '384', '3840'), ('123', '456', '7890')]
```

كما توقعنا، `re.findall` تنتج لنا مجموعه تحتوي على جميع ما يطابق نمط أرقام الهواتف.

RE.SUB

الدالة `re.sub(pattern, replacement, string)` تستبدل جميع تكرارات النمط المكتوب في المتغير `pattern` بالنص البديل في المتغير `replacement` في النص الموجود في المتغير `string`. هذه الدالة تعمل كما تعمل دالة `str.sub` في بايثون ولكن تستخدم التعبير النمطي.

في الكود البرمجي التالي، قمنا بتغير التواريخ لنحصل على تواريخ مطابقة بواسطة تبديل الفواصل إلى خطوط:

```
messy_dates = '03/12/2018, 03.13.18, 03/14/2018, 03:15:2018'
regex = r'[/.:]'
re.sub(regex, '-', messy_dates)
```

```
'03-12-2018, 03-13-18, 03-14-2018, 03-15-2018'
```

RE.SPLIT

الدالة `re.split(pattern, string)` تقوم بفصل النص المدخل في المتغير `string` في كل مرة يظهر فيها النمط في المتغير `pattern`. تعمل هذه الدالة بنفس طريقة عمل دالة `str.split` في بايثون ولكن تستخدم التعبير النمطي لإتمام عملية الفصل.

في الكود البرمجي التالي، استخدمنا `re. split` لفصل أسماء الفصول من أرقام الصفحات في جدول محتوى الكتاب:

```
```

```
toc = '''
PLAYING PILGRIMS============3
A MERRY CHRISTMAS===========13
THE LAURENCE BOY============31
BURDENS=====================55
BEING NEIGHBORLY============76
'''.strip()

# اولاً فصل كل سطر على حده
lines = re.split('\n', toc)
lines
```

```
['PLAYING PILGRIMS============3',
 'A MERRY CHRISTMAS===========13',
 'THE LAURENCE BOY============31',
 'BURDENS=====================55',
 'BEING NEIGHBORLY============76']
```

```
# ثم الفصل إلى اسم الفصل ورقم الصفحه
split_re = r'=+' # Matches any sequence of = characters
[re.split(split_re, line) for line in lines]
```

```
[['PLAYING PILGRIMS', '3'],
 ['A MERRY CHRISTMAS', '13'],
 ['THE LAURENCE BOY', '31'],
 ['BURDENS', '55'],
 ['BEING NEIGHBORLY', '76']]
```

التعابير النمطية و بانداز

تذكر أن مصفوفات بانداز تحتوي على دالة `.str` والتي تدعم دوال التعامل مع النصوص في بايثون. بنفس الطريقة، الدالة `.str` تدعم أيضا بعض الدوال من مكتبة `re`. سنشرح بعض الاستخدامات البسيطة للتعابير النمطية في بانداز وللمزيد من الدوال في الصفحة الرئيسية لدوال النصوص في بانداز.

قمنا بحفظ نص بسيط من أول خمس جمل من رواية *Little Women* في DataFrame. باستخدام دوال النصوص في بانداز يمكننا استخراج نصوص الحوارات المنطوقة في كل جمله:

```
text = '''
"Christmas won't be Christmas without any presents," grumbled Jo, lying on the rug.
"It's so dreadful to be poor!" sighed Meg, looking down at her old dress.
"I don't think it's fair for some girls to have plenty of pretty things, and other girls nothing at all,"
"We've got Father and Mother, and each other," said Beth contentedly from her corner.
The four young faces on which the firelight shone brightened at the cheerful words, but darkened again as
'''.strip()
little = pd.DataFrame({
    'sentences': text.split('\n')
})
```

```
little
```

	sentences
0	“Christmas won’t be Christmas without any pres…
1	“It’s so dreadful to be poor!” sighed Meg, loo…
2	“I don’t think it’s fair for some girls to hav…
3	“We’ve got Father and Mother, and each other,”…
4	The four young faces on which the firelight sh…

بما أن الحوارات المنطوقة تُكتب بين علامتي الاقتباس، سنقوم بكتابة تعبير نمطي بحث عن علامتي الاقتباس، ويوجد داخلها أي نص عدا علامتي اقتباس أخرى، ويجب أن ينتهي النص بعلامة اقتباس:

```
quote_re = r'"[^"]+"'
little['sentences'].str.findall(quote_re)
```

```
0    ["Christmas won't be Christmas without any pre...
1                     ["It's so dreadful to be poor!"]
2    ["I don't think it's fair for some girls to ha...
3     ["We've got Father and Mother, and each other,"]
4    ["We haven't got Father, and shall not have hi...
Name: sentences, dtype: object
```

بما أن الدالة `Series.str.findall` توجد لنا جميع التطابقات، بانداز توفر لنا دالتي `Series.str.extract` و `Series.str.extractall` لاستخراج التطابقات إلى مصفوفة أو DataFrame. هذه الدوال تشرط علينا أن يحتوي التعبير النمطي على الأقل علي مجموعه واحده:

```
# ايجاد النصوص بين علامتي اقتباس
quote_re = r'"([^"]+)"'
spoken = little['sentences'].str.extract(quote_re)
spoken
```

```
0    Christmas won't be Christmas without any prese...
1                        It's so dreadful to be poor!
2    I don't think it's fair for some girls to have...
3         We've got Father and Mother, and each other,
4    We haven't got Father, and shall not have him ...
Name: sentences, dtype: object
```

يمكننا إضافة ذلك كعمود في `little`:

```
little['dialog'] = spoken
little
```

	sentences	dialog
0	“Christmas won’t be Christmas without any pres…	Christmas won’t be Christmas without any prese…
1	“It’s so dreadful to be poor!” sighed Meg, loo…	It’s so dreadful to be poor!
2	“I don’t think it’s fair for some girls to hav…	I don’t think it’s fair for some girls to have…
3	“We’ve got Father and Mother, and each other,”…	We’ve got Father and Mother, and each other,
4	The four young faces on which the firelight sh…	We haven’t got Father, and shall not have him …

يمكننا أن نؤكد أن تلاعبنا بالنصوص باستخدام التعابير النمطية يعمل بالشكل الذي نرغب به في آخر جمله في ال DataFrame عن طريق طباعة النص الأصلي والنتيجة بعد المطابقة:

```
print(little.loc[4, 'sentences'])
```

```
The four young faces on which the firelight shone brightened at the cheerful words, but darkened again as
```

```
print(little.loc[4, 'dialog'])
```

```
We haven't got Father, and shall not have him for a long time.
```

ملخص التعابير النمطية في بايثون وبانداز

مكتبة `re` في بايثون توفر لنا الكثير من الدوال المفيدة للتلاعب بالنصوص باستخدام التعابير النمطية. عند العمل على DataFrame، نستخدم الدالة المماثلة للتلاعب بالنصوص باستخدام التعابير النمطية فيه بانداز.

لعرض كامل شرح مكتبة `re`، اضغط هنا.

لعرض كامل شرح دوال النصوص في بانداز، اضغط هنا.

قواعد البيانات العلائقية و SQL

مقدمة

حتى الآن، قمنا بالتعامل مع بيانات محفوظه في ملفات نصيه على الكمبيوتر. على الرغم من سهولة التعامل مع البيانات ذات الحجم الصغير في التحليل، إلا أن استخدام الملفات النصية لحفظها يظهر لنا بعض التحديات في حالات كثيره عندما نعمل على بيانات حقيقية.

الكثير من البيانات يتم جمعها من قبل أشخاص مختلفين، فريق من علماء البيانات مثلاً. إذا تم حفظ البيانات في ملفات نصيه، سيحتاج الفريق لتحميل وإرسال البيانات في كل مرة يتم التعديل عليها. الملفات النصية وحدها لا توفر طريقة مناسبه لإستعادتها ومشاركتها ليتم استخدامها بين عدد مختلف من المحللين. هذه المشكلة، إضافة إلى مشاكل أخرى، تجعل التعامل مع الملفات النصية كبيرة الحجم صعباً خاصه بين فريق من المحللين.

عادةً ما ننتقل لأنظمة قواعد البيانات العلائقية (RDBMS) لحفظ البيانات، مثلاً MySQL أو PostgreSQL. للعمل على هذه الأنظمة، نستخدم لغة الاستعلام SQL بدلاً من بايثون. في هذا الفصل، سنتحدث عن قواعد البيانات العلائقية ونتعرف على SQL.

النموذج العلائقي

قاعدة البيانات هي مجموعة مُرتبه من البيانات. في السابق، كانت البيانات محفوظه ومصممه بشكل معين لتفي بغرض ما. مثلاً، شركة طيران قد تقوم بحفظ معلومات رحلة بطريقه مختلفة عن حفظ بنك لمعلومات حساب. في عام 1969، قام تيد كود Ted Codd بتعريف النموذج العلائقي كطريقه عامه لحفظ البيانات. البيانات محفوظه في جداول ثنائية الأبعاد تسمى **علاقات Relations**، تحتوي على ملاحظات معينة في كل صف (عادة ما تعرف بمصفوفات Tuples). كل مصفوفة تحتوي على بعض **السمات Attributes** والتي تصف العلاقات بين الجداول. كل سمه عن علاقة لديها اسم ونوع.

لنأخذ جدول العلاقات التالي `purchases`:

date purchased	retailer	product	name
Jun-16-03	Best Buy	iPod	Samantha
Jul-16-08	Amazon	Chromebook	Timothy
Oct-16-02	Target	Surface Pro	Jason

في `purchases`، كل صف يصف علاقة بين السمات `retailer`، `product`، `name` و `date purchased`.

مخطط العلاقات Relation schema تحتوي على أسماء الأعمدة، أنواع البيانات وقيودها. مثلاً، مخطط جدول `purchases` يحدد أن هناك الأعمدة `retailer`، `product`، `name` و `date purchased`، ويوضح أيضاً أن كل عمود يحتوي على نص.

جدول العلاقات التالي `prices` يوضح السعر الحالي لمنتجات في أسواق مختلفة:

price	product	retailer
719.00	Galaxy S9	Best Buy
200.00	iPod	Best Buy
450.00	iPad	Amazon
24.87	Battery pack	Amazon
249.99	Chromebook	Amazon
215.00	iPod	Target
799.00	Surface Pro	Target
659.00	Google Pixel 2	Target
238.79	Chromebook	Walmart

الآن يمكننا العودة لكلا الجدولين لتحديد كم دفع كل من Timothy، Samantha، و Jason لمنتجاتهم (لنفترض بقاء الأسعار كما هي في جميع الأسواق والأوقات). معاً، كلا الجدولين يكونا **قاعدة بيانات علائقية**، والتي هي عبارة عن جداول بينها أكثر من علاقة. المخطط لكامل قاعدة البيانات هي مصفوفة تحتوي على جميع مخططات العلاقات في قاعدة البيانات.

أنظمة قواعد البيانات العلائقية

يمكننا وصف قواعد البيانات العلائقية بأنها مجموعه من الجداول تحتوي على صفوف لمدخلات. نظام قاعدة بيانات علائقية (RDBMS) يوفر واجهه للمستخدم لتلك القواعد. MySQL، Oracle و PostgreSQL هي ثلاث من أكثر قواعد البيانات العلائقية إستخداماً.

توفر هذه الأنظمة المستخدم إمكانية إضافة، تعديل، وحذف البيانات منها. إضافة لعدد من المميزات لتميزها عن استخدام الملفات النصية لحفظ البيانات، وهي:

- ضمانية حفظ البيانات: تحمي هذه الأنظمة البيانات من مشاكل الأنظمة المختلفة.
- الأداء: تحفظ البيانات بشكل أكثر كفاءة من الملفات النصية ولديها خوارزميات صُممت بشكل متقن للاستعلام عن البيانات.
- إدارة البيانات: توفر أدوات للتحكم بالوصول للبيانات، للحماية من المستخدمين غير المصرح لهم بالوصول للبيانات الحساسة.
- تناسق البيانات: يمكن لهذه الأنظمة فرض قيود على المحتوى المُدخل، مثلاً، العمود `GPA` يجب أن يحتوي فقط على رقم عشري من 0. 0 حتى 4. 0.

للتعامل مع البيانات المحفوظة فيه قواعد RDBMS نستخدم SQL.

RDBMS × بانداز

ما الفرق بين RDBMS وبانداز أولاً، بانداز لا تعتبر وسيلة لحفظ البيانات. على الرغم أن ال DataFrames في بانداز يمكنها الكتابة والقراءة من عدة أشكال للبيانات، لا يمكن لبانداز التحكم بطريقة حفظ البيانات في الكمبيوتر كما في RDBMS. ثانياً، بانداز توفر بشكل أساسي أدوات للتعامل مع البيانات، بينما RDBMS توفر طرق حفظ والتعامل مع البيانات معاً، مما يجعلها خياراً أفضل للبيانات ذات الحجم الكبير. من القواعد العامة هي أن نستخدم RDBMS عندنا تكون حجم البيانات لدينا أكثر من عدة قيقا بايت. أخيرا، للتعامل مع بانداز نحتاج لمعرفة بايثون، بينما في RDBMS نحتاج لمعرفة SQL. وبما أن SQL أسهل بكثير للتعلم من بايثون، مما يجعل RDBMS سهله للتعامل من قبل المستخدمين غير التقنيين.

SQL

SQL (لغة الاستعلام الهيكلية) هي لغة برمجه لديها عمليات لتحديد، ترتيب، تعديل وإجراء العمليات الحسابية على بيانات محفوظه في أنظمة قواعد البيانات العلائقية (RDBMS).

تعتبر SQL لغة برمجه تعريفية. يعني ذلك أن المستخدم يجب أن يحدد أي نوع من البيانات يحتاج *what*، وليس كيف يحصل عليه *how*. للتوضيح:

- **Declarative تعرفيه**: قم بعملية حساب للأعمدة x و y من الجدول A عندما تكون القيمة في العمود y أكبر من 100.00.
- **Imperative أمرية**: لكل قيمه في الجدول A، تحقق أن كانت القيمة للمتغير y أكثر من 100. إذا كانت كذلك، احفظ سمات الأعمدة x و y في جدول جديد. وأظهر لنا الجدول الجديد.

في هذا الفصل، سنكتب استعلامات SQL كنصوص في بايثون، ثم نستخدم بانداز لتنفيذ هذه أوامر الاستعلام وقراءة النتائج ك DataFrame في بانداز أثناء شرحنا للأوامر في SQL، سنظهر طريقة كتابتها في بانداز للمقارنة بينهما.

تنفيذ أوامر الاستعلام في بانداز

لتنفيذ أوامر استعلام SQL في بايثون، سنقوم بالتواصل مع قاعدة البيانات باستخدام مكتبة sqlalchemy. يمكننا لاحقاً استخدام الدالة pd.read_sql لتنفيذ أوامر SQL.

لتحميل قاعدة البيانات sql_basics.db اضغط هنا.

```
import sqlalchemy

# pd.read_sql يقبل متغير engine يوصلنا بقاعدة البيانات، والذي انشأناه بالأسفل
sqlite_uri = "sqlite:///sql_basics.db"
sqlite_engine = sqlalchemy.create_engine(sqlite_uri)
```

تحتوي قاعدة البيانات على جدول علاقات واحد: `prices`. لإظهاره نستخدم أمر استعلام في SQL باستخدام الدالة `read_sql` على قاعدة البيانات العلائقية RDBMS، ثم تأتينا النتيجة ك DataFrame:

```
sql_expr = """
SELECT *
FROM prices
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	product	price
0	Best Buy	Galaxy S9	719
1	Best Buy	iPod	200
2	Amazon	iPad	450
3	Amazon	Battery pack	24.87
4	Amazon	Chromebook	249.99
5	Target	iPod	215
6	Target	Surface Pro	799
7	Target	Google Pixel 2	659
8	Walmart	Chromebook	238.79

لاحقاً في هذا الفصل، سنقارن بين استعلامات SQL و دوال بانداز لذا قمنا بكتابة DataFrame تحتوي على نفس المحتوى السابق وحفظناها في المتغير `prices`:

```
import pandas as pd

prices = pd.DataFrame([['Best Buy', 'Galaxy S9', 719.00],
                       ['Best Buy', 'iPod', 200.00],
                       ['Amazon', 'iPad', 450.00],
                       ['Amazon', 'Battery pack', 24.87],
                       ['Amazon', 'Chromebook', 249.99],
                       ['Target', 'iPod', 215.00],
```

```
                    ['Target', 'iPod', 215.00],
                    ['Target', 'Surface Pro', 799.00],
                    ['Target', 'Google Pixel 2', 659.00],
                    ['Walmart', 'Chromebook', 238.79]],
                   columns=['retailer', 'product', 'price'])
```

	retailer	product	price
0	Best Buy	Galaxy S9	719
1	Best Buy	iPod	200
2	Amazon	iPad	450
3	Amazon	Battery pack	24.87
4	Amazon	Chromebook	249.99
5	Target	iPod	215
6	Target	Surface Pro	799
7	Target	Google Pixel 2	659
8	Walmart	Chromebook	238.79

طريقة كتابة أوامر SQL

جميع أوامر الاستعلام في SQL تكون بهذا الشكل:

```
SELECT [DISTINCT] <column expression list>
FROM <relation>
[WHERE <predicate>]
[GROUP BY <column list>]
[HAVING <predicate>]
[ORDER BY <column list>]
[LIMIT <number>]
```

لاحظ أن:

- **جميع ما في [داخل الأقواس] هي أوامر اختياريه.** ليكون أمر الاستعلام صحيح يجب أن يحتوي على SELECT و FROM.
- **في العادة تكتب أوامر الاستعلام في SQL بأحرف إنجليزية كبيرة.** على الرغم أن تكبير الأحرف ليس مطلوب، لكن يعتبر من الطرق المتبعة الصحيحة لكتابة الأوامر، لتسهل على المستخدمين الآخرين قراءة استعلاماتك.
- في الأمر FROM يمكننا الاستعلام عن أكثر من جدول، ولكن في هذا الفصل سنتدرب على استخدام جدول واحد لجعل الأمر سهلاً على المتعلم.

SELECT و FROM

الأمران الإلزاميان في استعلامات SQL هي:

- SELECT وتعني الأعمدة التي نريد إظهارها.
- FROM ويقصد بها الجداول التي نأخذ منها الأعمدة.

لعرض جميع محتوى جدول prices، نطبق الأمر:

```
sql_expr = """
SELECT *
FROM prices
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	product	price
0	Best Buy	Galaxy S9	719
1	Best Buy	iPod	200
2	Amazon	iPad	450
3	Amazon	Battery pack	24.87
4	Amazon	Chromebook	249.99
5	Target	iPod	215
6	Target	Surface Pro	799
7	Target	Google Pixel 2	659
8	Walmart	Chromebook	238.79

الاستعلام SELECT * يجلب لنا جميع الأعمدة. لعرض فقط عمود retailer، نقوم بإضافتها كالتالي:

```
sql_expr = """
SELECT retailer
FROM prices
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer
0	Best Buy
1	Best Buy
2	Amazon
3	Amazon
4	Amazon
5	Target
6	Target
7	Target
8	Walmart

إذا أردنا عرض القيم في الجدول دون تكرار، نقوم بإضافة DISTINCT:

```
sql_expr = """
SELECT DISTINCT(retailer)
FROM prices
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer
0	Best Buy
1	Amazon
2	Target
3	Walmart

طريقة كتابة هذا الاستعلام في باندا تكون كالتالي:

```
prices['retailer'].unique()
```

```
array(['Best Buy', 'Amazon', 'Target', 'Walmart'], dtype=object)
```

كل نظام قواعد بيانات يأتي بدوالة الخاصة التي يمكن تطبيقها، مثلاً دوال مقارنة، العمليات الحسابية، ودوال النصوص. في هذه المادة نستخدم PostgreSQL، نظام قواعد بيانات علائقي ويأتي معه بالكثير من الدوال. يمكن تصفح كامل الدوال هنا. تذكر دائماً أن كل نظام لديه دوالة الخاصة.

الاستعلام التالي يحول كل أسماء الأسواق إلى حروف كبيرة، ويقوم بقسمة سعر الشراء على 2:

```
sql_expr = """
SELECT
    UPPER(retailer) AS retailer_caps,
    product,
    price / 2 AS half_price
FROM prices
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer_caps	product	half_price
0	BEST BUY	Galaxy S9	359.5
1	BEST BUY	iPod	100
2	AMAZON	iPad	225
3	AMAZON	Battery pack	12.435
4	AMAZON	Chromebook	124.995
5	TARGET	iPod	107.5
6	TARGET	Surface Pro	399.5

	retailer_caps	product	half_price
7	TARGET	Google Pixel 2	329.5
8	WALMART	Chromebook	119.395

لاحظ أن بإمكاننا استخدام مسميات مستعارة للجداول **Alias** باستخدام AS لكي يظهر العمود بمسمى جديد. لكن هذا لا يغير شيء باسم العمود الأصلي

WHERE

WHERE تُمكنا من تحديد شروط معينة للبيانات. مثلاً، إذا أردنا إيجاد فقط المنتجات أقل من $500:

```
sql_expr = """
SELECT *
FROM prices
WHERE price < 500
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	product	price
0	Best Buy	iPod	200
1	Amazon	iPad	450
2	Amazon	Battery pack	24.87
3	Amazon	Chromebook	249.99
4	Target	iPod	215
5	Walmart	Chromebook	238.79

يمكننا أيضا استخدام OR، AND و NOT لتحديد استعلامنا بشكل أدق. مثلاً، للبحث عن منتجات Amazon والتي لا تحتوي على Battery pack وسعرها أقل من $300، نكتب الاستعلام التالي:

```
sql_expr = """
SELECT *
FROM prices
WHERE retailer = 'Amazon'
    AND NOT product = 'Battery pack'
    AND price < 300
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	product	price
0	Amazon	Chromebook	249.99

ولكتابة نفس الأمر في بانداز:

```
prices[(prices['retailer'] == 'Amazon')
   & ~(prices['product'] == 'Battery pack')
   & (prices['price'] <= 300)]
```

	retailer	product	price
4	Amazon	Chromebook	249.99

نلاحظ هناك وجود فرق يجب التنبيه عنه، الرقم التسلسلي للمنتجي Chromebook في استعلام SQL كان 0، ولكن في استعلام بايثون 4. هذا لأن استعلامات SQL تقوم دائماً بعرض البيانات في جدول جديد وبأرقام تسلسليه جديده تبدأ من 0، ولكن في بانداز تقوم بعرض جزء من ال DataFrame مع استخدام نفس أرقام التسلسل. يمكننا استخدام الدالة pd.DataFrame.reset_index لإعادة تعين الأرقام التسلسلية في بانداز

دوال الجمع

حتى الآن، تعاملنا مع بيانات موجودة في الجداول، يعني ذلك، كل النتائج التي ظهرت سابقاً هي أجزاء من المدخلات في الجداول. ولكن لتطبيق تحليل للبيانات، نحتاج للقيام ببعض عمليات التجميع على بياناتنا. في SQL، يطلق عليها **دوال التجميع Aggregate Functions**.

لنوجد قيمة متوسط أسعار جميع المنتجات في جدول prices:

```
sql_expr = """
SELECT AVG(price) AS avg_price
FROM prices
```

```
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	avg_price
0	395.072222

وفي بانداز نكتبها:

```
prices['price'].mean()
```

```
395.0722222222222
```

قائمة كاملة بجميع دوال التجميع في PostgreSQL هنا. على الرقم من أننا نستخدمها كأداة أساسيه للتعامل مع SQL في هذه المادة، تذكر أن هناك أنواع مختلفة من SQL مثل SQLite، MySQL وغيرها التي تستخدم أسماء مختلفة للدوال وبعض الأحيان تحتوي على دوال مختلفة.

GROUP BY و HAVING

باستخدام دوال التجميع، يمكننا القيام باستعلامات مُعقده. لاستخدام دوال تجميع سهله التعامل، يمكننا تجربة التالية:

- GROUP BY تستقبل أسماء أعمدة وتقوم بجمعها كما في دالة pd.DataFrame.groupby في بانداز.
- HAVING تعمل بشكل مشابه ل WHERE1، ولكن تستخدم فقط على البيانات الناتجة ممن الدوال المجمعه. (ملاحظة: لاستخدام HAVING، يجب أن تكون مسبوقة ب GROUP BY)

**ملاحظة مهمة: ** عند استخدام GROUP BY، يجب أن تكون جميع الأعمدة في SELECT موجودة في GROUP BY أو تكون دالة تجميع مطبقه عليها.

يمكننا استخدام الاستعلام التالي لإيجاد أعلى سعر لكل متجر:

```
sql_expr = """
SELECT retailer, MAX(price) as max_price
FROM prices
GROUP BY retailer
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	max_price
0	Amazon	450
1	Best Buy	719
2	Target	799
3	Walmart	238.79

لنقل أن لدينا عميل ذو ذوق عالي ويريد فقط تلك المتاجر التي تبيع منتجات بسعر أعلى من $700. لاحظ أن يجب علينا استخدام HAVING لإيجاد النتيجة من عمود تم تجميعه؛ لا يمكننا استخدام WHERE لفلترة نتائج عمود مُجَمع. لإيجاد قائمة بالمتاجر التي تبيع المنتجات المفضلة لعميلنا نقوم بالتالي:

```
sql_expr = """
SELECT retailer, MAX(price) as max_price
FROM prices
GROUP BY retailer
HAVING max_price > 700
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	max_price
0	Best Buy	719
1	Target	799

وللمقارنة مع بايثون:

```
max_prices = prices.groupby('retailer').max()
max_prices.loc[max_prices['price'] > 700, ['price']]
```

	price
retailer	

	price
Best Buy	719
Target	799

ORDER BY و LIMIT

تُمكنا الأوامر التالية من التحكم بكيفية عرض البيانات:

- `ORDER BY` تساعدنا على عرض البيانات بشكل مُرتب بِناءًا على القيم داخل العمود بشكل تلقائي، `ORDER BY` تستخدم الترتيب التصاعدي `ASC` ولكن يمكننا عكسها وعرض البيانات بشكل تنازلي باستخدام `DESC`.
- `LIMIT` تسمح لنا بتحديد كم من البيانات نعرض.

لنقم بعرض أقل المنتجات سعراً في جدول `prices`:

```
sql_expr = """
SELECT *
FROM prices
ORDER BY price ASC
LIMIT 3
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	retailer	product	price
0	Amazon	Battery pack	24.87
1	Best Buy	iPod	200
2	Target	iPod	215

لاحظ أننا لم نحتاج لإضافة `ASC` لأن `ORDER BY` تجلب لنا البيانات بترتيب تصاعدي بشكل تلقائي. للمقارنة مع باندا:

```
prices.sort_values('price').head(3)
```

	retailer	product	price
3	Amazon	Battery pack	24.87
1	Best Buy	iPod	200
5	Target	iPod	215

مره أخرى، نلاحظ أن الأرقام التسلسليه غير مرتبه في باندا. كما في السابق، باندا تقوم بعرض DataFrame كما هي، عل عكس SQL التي تنشأ جدول جديد عند كُل استعلام.

ترتيب الأوامر في SQL

تطبق الأوامر في SQL بشكل مُرتب. للأسف، أن الترتيب مختلف عن ما يظهر في كتابة الاستعلام. ترتيب الأوامر كالتالي:

- `FROM`: جدول أو أكثر من جدول.
- `WHERE`: تطبيق شروط على الأسطر.
- `GROUP BY`: التجميع.
- `HAVING`: تطبيق الشروط على التجميع.
- `SELECT`: اختيار الأعمدة

ملاحظة في `WHERE` و `HAVING`: بما أن `WHERE` تُطبق قبل `GROUP BY`، فإن `WHERE` لا تفيدنا عندما نقوم بالتجميع. لذا، للتعامل مع بيانات مجمعه نستخدم `HAVING`.

ملخص SQL

في هذا الجزء قمنا بشرح طريقة كتابة استعلامات SQL واهم الأوامر لإجراء تحليل للبيانات باستخدام أنظمة قواعد البيانات العلائقية.

الربط في SQL

في باندا يمكننا استخدام الدالة `pd.merge` لجمع جدولين معاً يحتويان على قيم متشابهه في أحد الأعمدة. مثلاً:

```
pd.merge(table1, table2, on='common_column')
```

في هذا الجزء، سنتعلم الجمع فيه SQL والذي يفيدنا في جمع أكثر من جدول في قواعد البيانات العلائقية.

لنفرض مثلاً أننا متجر لمنتجات القطط ولدينا قاعدة بيانات بالقطط التي في متجرنا. لدينا **جدولين** مختلفين: `names` و `colors`. جدول `names` يحتوي على الأعمدة:

- `cat_id`: رقم مميز لكل قط.
- `name`: اسم القط.

جدول `colors` يحتوي على الأعمدة:

- `cat_id`: رقم مميز لكل قط.
- `color`: لون كل قط.

لاحظ وجود قيم مفقوده في كل جدول. القيمة 3 في العمود `cat_id` مفقوده من جدول `names`، والقيمة 4 من عمود `cat_id` مفقوده في جدول `colors`

Colors Table

color	cat_id
orange	0
black	1
calico	2
white	3

Names Table

name	cat_id
Apricot	0
Boots	1
Cally	2
Eugene	4

لإيجاد لون القط ذو الاسم Apricot، يجب علينا استخدام معلومه من جدولين. يمكننا ربط الجدولين باستخدام العمود `cat_id`، ويكون لدينا جدول جديد يحتوي على `name` و `color`.

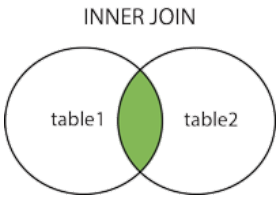
الربط

الربط بين عدة جداول باستخدام قيم في أعمدتها.

يوجد أربع أنواع من الربط: الربط الداخلي Inner Join, الربط الخارجي Outer Join أو الكامل Full Join, الربط اليميني Right Join والربط اليساري Left Join. بالرغم بأن جميعها تربط بين الجداول وتقوم بجمعها، إلا أن كل نوع يعامل القيم بطريقة مختلفة.

Inner Join

تعريف: في الربط الداخلي، الجدول النهائي يحتوي على **القيم المتشابهة** في الجدولين المربوطين معاً.



مثال: نريد الربط بين جدولي `names` و `colors` لجمع كل قطه مع لونها. بما أن كلا الجدولين يحتويان على العمود `cat_id` وهو رقم مميز لكل قط، يمكننا استخدام الربط الداخلي باستخدام العمود `cat_id`.

SQL: لكتابة الربط الداخلي في SQL، نقوم بتعديل `FROM` بإضافة `INNER JOIN` عليها:

```
SELECT ...
FROM <TABLE_1>
    INNER JOIN <TABLE_2>
    ON <...>
```

مثلاً:

```
SELECT *
FROM names AS N
    INNER JOIN colors AS C
    ON N.cat_id = C.cat_id;
```

	cat_id	name	cat_id	color
0	0	Apricot	0	orange
1	1	Boots	1	black
2	2	Cally	2	calico

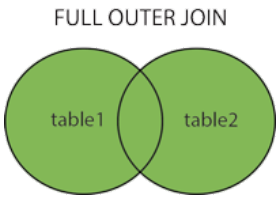
يمكنك التحقق من أن كل اسم قطه حصل على لونها الصحيح. لاحظ، القطة صاحبة cat_id رقم 3 و 4 لم يتم جمعها في الجدول النهائي لأن جدول colors لا يحتوي على سطر بقيمة cat_id تساوي 4 و جدول names لا يحتوي على سطر cat_id بقيمة 3. في الدمج الداخلي، إذا لم يكون هناك قيم متشابهه في كلا الجدولين، لن يتم ضم القيمة في النتيجه النهائية.

لنفترض أن لدينا DataFrame بإسم names وأخرى بإسم colors، يمكننا كتابة الدمج الداخلي في بانداز بالطريقة التالية:

```
pd.merge(names, colors, how='inner', on='cat_id')
```

Full/Outer Join

تعريف: في الربط الكامل/الخارجي، **كل القيم في كلا الجدولين** يتم أضافتها في الجدول النهائي. إذا كانت القيمة موجودة في جدول دون الآخر، فيتم تعويضها ب NULL.



مثال: كما في السابق، نريد الربط بين الجدول names و colors لإعطاء كل قطه لونها. هذه المرة، نريد إظهار جميع القيم في الجداول حتى لو لا يجود لها قيم مُطابقه.

SQL: لكتابة الربط الكامل/الخارجي في SQL، نقوم بتعديل FROM بإضافة FULL JOIN عليها:

```
SELECT ...
FROM <TABLE_1>
    FULL JOIN <TABLE_2>
    ON <...>
```

مثلاً:

```
SELECT name, color
FROM names N
    FULL JOIN colors C
    ON N.cat_id = C.cat_id;
```

cat_id	name	color
0	Apricot	orange
1	Boots	black
2	Cally	calico
3	NULL	white
4	Eugene	NULL

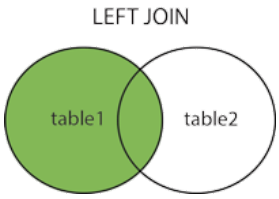
لاحظ في النتيجة النهاية ظهور القيم 3 و 4 في عمود cat_id. إذا كانت أحد القيم تظهر في جدول دون الآخر، فيتم إضافتها للجدول النهائي مع القيمة NULL بدلاً من القيم المفقودة.

في بانداز نكتبها كالتالي:

```
pd.merge(names, colors, how='outer', on='cat_id')
```

Left Join

تعريف: جميع القيم في الجدول على اليسار يتم ربطها في الجدول النهائي. إذا كانت قيمه في الجدول اليساري لا توجد لها مطابق على الجدول الآخر، يتم تعويض قيمتها المفقودة ب NULL.



مثال: نربط الجدولان `names` و `colors` لإظهار ألوان القطط. هنا، نريد إظهار جميع أسماء القطط حتى لو لم يكن لها لون في الجدول `colors`.

SQL: لكتابة الربط اليساري في SQL، نقوم بتعديل `FROM` بإضافة `LEFT JOIN` عليها:

```
SELECT ...
FROM <TABLE_1>
    LEFT JOIN <TABLE_2>
    ON <...>
```

مثلاً:

```
SELECT name, color
FROM names N
    LEFT JOIN colors C
    ON N.cat_id = C.cat_id;
```

cat_id	name	color
0	Apricot	orange
1	Boots	black
2	Cally	calico
4	Eugene	NULL

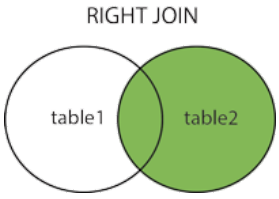
لاحظ أن الجدول النهائي يحتوي على جميع أسماء القطط. ثلاثة من القيم `cat_id` في جدول `names` تحتوي على قيم بنفس ال `cat_id` في جدول `colors` وواحده لا تحتوي على قيمه مطابقة (Eugene). القط الذي لا يوجد له لون تم تعويض قيمته ب `NULL`.

في باندا نكتب الاستعلام كالتالي:

```
pd.merge(names, colors, how='left', on='cat_id')
```

Right Join

تعريف: جميع القيم في الجدول على اليمين يتم ربطها في الجدول النهائي. إذا كانت قيمه في الجدول اليميني لا توجد لها مطابق على الجدول الآخر، يتم تعويض قيمتها المفقودة ب `NULL`.



مثال: الجدولان `names` و `colors`، نريد ربطها كل قط بلونه. لكن، نريد إظهار جميع ألوان القطط حتى لو لم يكن لها أسماء.

SQL: لكتابة الربط اليمين في SQL، نقوم بتعديل `FROM` بإضافة `RIGHT JOIN` عليها:

```
SELECT ...
FROM <TABLE_1>
    RIGHT JOIN <TABLE_2>
    ON <...>
```

مثلاً:

```
ELECT name, color
FROM names N
```

```
    RIGHT JOIN colors C
    ON N.cat_id = C.cat_id;
```

cat_id	name	color
0	Apricot	orange
1	Boots	black
2	Cally	calico
3	NULL	white

هذه المرة، الجدول النهائي يظهر جميع الألوان. ثلاثة من القيم في العمود `cat_id` بجدول `colors` لديها مطابق في عمود `cat_id` بجدول `names` وقيمه واحد لم تجد مطابق (white). اللون الذي لم يجد له اسم مطابق سيعوض بالقيمة `NULL`.

في باندا:

```
pd.merge(names, colors, how='right', on='cat_id')
```

Implicit Inner Join

عادةً توجد أكثر من طريقه لإيجاد نتيجة معينة في SQL تماما كما في بايثون يوجد أكثر من طريقه لحل مشاكل. سنوضح طريقه أخرى لكتابة استعلامات Inner Join تسمى *Implicit Join*.

كتبنا سابقاً الربط الداخلي التالي:

```
SELECT *
FROM names AS N
    INNER JOIN colors AS C
    ON N.cat_id = C.cat_id;
```

الطريقة الأخرى *Implicit* لكتابة هذا الأمر يكون شكلها مختلف ودون `INNER JOIN`. لاحظ أن `FORM` يستخدم فيها الفواصل بين جدولين والشرط `WHERE` يحدد شروط الربط:

```
SELECT *
FROM names AS N, colors AS C
WHERE N.cat_id = C.cat_id;
```

عند استخدام أكثر من جدول في `FORM`، تقوم SQL بإنشاء جدول يحتوي على كل سطر بكلا الجدولين، مثلاً:

```
sql_expr = """
SELECT *
FROM names N, colors C
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	cat_id	name	cat_id	color
0	0	Apricot	0	orange
1	0	Apricot	1	black
2	0	Apricot	2	calico
3	0	Apricot	3	white
4	1	Boots	0	orange
5	1	Boots	1	black
6	1	Boots	2	calico
7	1	Boots	3	white
8	2	Cally	0	orange
9	2	Cally	1	black
10	2	Cally	2	calico
11	2	Cally	3	white
12	4	Eugene	0	orange
13	4	Eugene	1	black
14	4	Eugene	2	calico

	cat_id	name	cat_id	color
15	4	Eugene	3	white

يطلق على هذه العملية *بالضرب الديكارتي Cartesian product*، كل سطر في الجدول الأول يربط بكل سطر في الجدول الثاني. لاحظ أن الكثير من الأسطر تحتوي على ألوان قطط لا تتطابق مع أسمائها. الشرط الإضافي في `WHERE` يطبق الربط ويفلتر القيم التي لا تتطابق في عمود `cat_id`:

```
SELECT *
FROM names AS N, colors AS C
WHERE N.cat_id = C.cat_id;
```

	cat_id	name	cat_id	color
0	0	Apricot	0	orange
1	1	Boots	1	black
2	2	Cally	2	calico

ربط أكثر من جدول

لربط أكثر من جدول، نضيف على المتغير `FROM` أوامر الربط `JOIN`. مثلاً، جدول الأعمار للقطط:

age	cat_id
4	0
3	1
9	2
20	4

لإجراء عملية الربط الداخلي على الجداول `colors`، `neams` و `ages` نكتب التالي:

```
sql_expr = """
SELECT name, color, age
    FROM names n
    INNER JOIN colors c ON n.cat_id = c.cat_id
    INNER JOIN ages a ON n.cat_id = a.cat_id;
"""
pd.read_sql(sql_expr, sqlite_engine)
```

	name	color	age
0	Apricot	orange	4
1	Boots	black	3
2	Cally	calico	9

ملخص الربط

قمنا بعرض الأربع أنواع الأساسيه للربط في SQL: الربط الداخلي، الكامل، اليميني واليساري. نستخدم جميع الأنواع في الربط للقيام بعمليات الربط بين الجداول المنفصله التي تربطها علاقات، كل عملية ربط لديها طرق مختلفة للربط بين القيم.

النماذج والتوقعات

مقدمة

> بشكل عام، جميع النماذج خاطئة، ولكن بعضها مفيد.
>
> — جورج بوكس، إحصائي

تحدثنا عن تكوين السؤال، تنظيف البيانات والتحليل الاستكشافي للبيانات، الخطوات الثلاث الأولى في دورة حياة علم البيانات. ورأينا أن التحليل الاستكشافي يظهر علاقات بين القيم في البيانات. كيف يمكننا التحقق إذا كانت هذه العلاقة حقيقة أو لا؟ كيف يمكننا استخدام هذه العلاقات للحصول على توقعات عن المستقبل؟ للإجابة على هذه الأسئلة سنحتاج لأداة رياضية لبناء النماذج والتوقعات.

النموذج Model هو عبارة عن أفضل تمثيل لنظامنا. مثلاً، إذا قمنا برمي كره من الفولاذ من فوق برج بيزا المائل، نموذج بسيط للجاذبية سيتوقع أن الكره سترتطم بالأرض بسرعة 9. 8 م/ث². هذا النموذج سمح لنا بتوقع كم ستأخذ الكرة من وقت للوصول إلى الأرض باستخدام قوانين الحركة.

هذا النموذج للجاذبية يوضح سلوك وأداء نظامنا ولكن فقط في التوقع، لا يقوم بحساب تأثير حركة الهواء، والطفو في الهواء والتأثيرات الأخرى على جاذبية الأجسام أثناء السقوط. بسبب هذه العوامل غير المدروسة، نموذجنا سينتج دائماً توقعات خاطئة! ولكن، النموذج البسيط للجاذبية يعتبر دقيق بشكل كافي في كثير من الأحيان مما يجعله يستخدم ويُعلم اليوم.

وبنفس الطريقة، كل نموذج نقوم بتعريفه باستخدام بيانات يعتبر تقدير لما هو بالواقع. عندما لا يكون التقدير دقيق جداً، نموذجنا يكون استخدامه مفيد وعملي. وبطبيعة الحال، ستظهر لنا أسئلة أساسيه مثل، كيف نختار النموذج؟ كيف نعرف ما إذا كنا نحتاج نموذج أكثر تعقيداً؟

في الفصول القادمة من هذا الكتاب، سنقوم بتطوير أدوات حسابية لتصميم وندرب النموذج على البيانات. سنقوم أيضا بالتعرف على أدوات استنتاجية تُمكنا من معرفة إذا كان بإمكاننا تعميم نماذجنا.

النماذج

في الولايات المتحدة، الكثير من زوار المطاعم يتركون بعض من المال للنادل أثناء دفعهم لثمن وجباتهم. رغم أنه من المعروف في الولايات المتحدة أن 15% من مجموع الفاتورة يكون كإكرامية، إلا أن بعض المطاعم لديها زبائن مخلصين أكثر من غيرها يقدمون المزيد من المال كإكرامية.

أحد النوادل المهتمين أراد معرفة ما قد يحصل عليه من إكرامية فقام بجمع معلومات عن كافة الطاولات التي خدمها في غضون شهر:

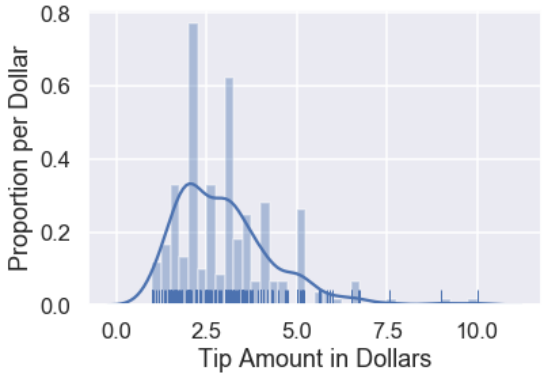
```
import seaborn as sns

tips = sns.load_dataset('tips')
tips
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.5	Male	No	Sun	Dinner	3
...	...	...	...	...	...	...	...
241	22.67	2	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3	Female	No	Thur	Dinner	2

244 rows × 7 columns

```
sns.distplot(tips['tip'], bins=np.arange(0, 10.1, 0.25), rug=True)
plt.xlabel('Tip Amount in Dollars')
plt.ylabel('Proportion per Dollar');
```

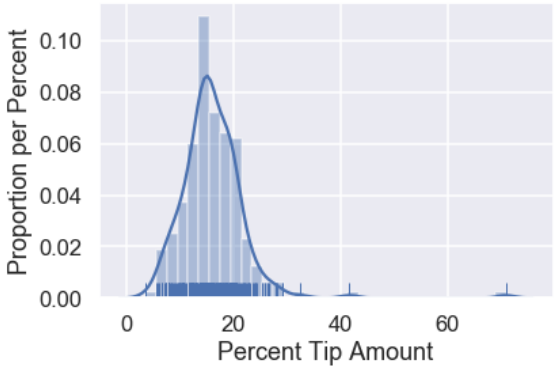
يمكننا رسم مدرج تكراري لمجموع الإكراميات:



بإمكاننا ملاحظة بعض الأنماط في البيانات. مثلاً، هناك تكرار كبير للقيمة 2$ وغالب الإكراميات من مضاعفات 0.50$.

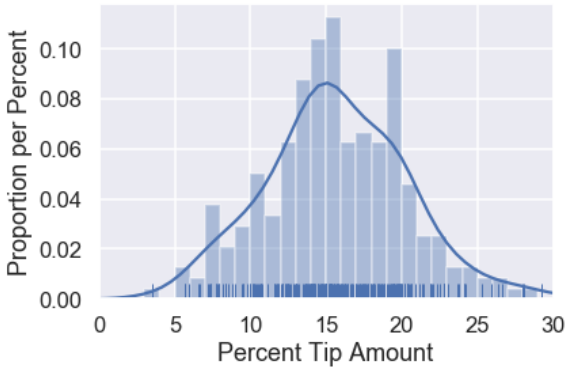
حالياً، نحن مهتمين بنسبة الإكرامية: قيمة الإكرامية قسمة مجموع الفاتوره. يمكننا إنشاء عمود جديد في ال DataFrame لحفظ هذه القيمه ونعرض توزيع النتائج:

```
tips['pcttip'] = tips['tip'] / tips['total_bill'] * 100
sns.distplot(tips['pcttip'], rug=True)
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



يبدو أن أحد الزبائن ترك للنادل إكرامية قيمتها 70%! ولكن أغلب الإكراميات كان أقل من 30%. لنقرب الرسم البياني إلى ذلك الجزء:

```
sns.distplot(tips['pcttip'], bins=np.arange(30), rug=True)
plt.xlim(0, 30)
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



نلاحظ أن توزيع البيانات في الغالب حول 15% مع مجموع تكرار أعلى حول 20%. لنفترض أن نادلنا يريد معرفة كم النسبة التي سيحصل عليها من الزبائن. للإجابة على السؤال، يمكننا بناء نموذج يتوقع كم مبلغ الإكرامية الذي سيحصل عليها النادل.

النموذج البسيط

أحد أمثله على النموذج البسيط هي تجاهل كل البيانات وذكر أنه من المعروف في الولايات المتحدة إعطاء إكرامية بمقدار 15% من مبلغ الفاتورة، بذلك يحصل النادل دائماً على 15% من كل زبون. على الرغم من بساطته، سنستخدم هذه النموذج للتعريف ببعض المتغيرات لاحقاً.

هذا النموذج يفترض أن هناك نسبه وحيده صحيحه لكل الإكراميات التي سيحصل عليها النادل من كل زبائنه. هذه هي *معالم المجتمع Population Parameter* لنسبة الإكرامية، والتي سنرمز لها بالرمز θ^*.

بعد فرضيتنا السابقة، نموذجنا سيقول إن توقعه لقيمة θ^* هي 15%. سنستخدم الرمز θ لوصف التوقع (التنبؤ).

برموز رياضية، فإن نموذجنا يفرض التالي:

$$\theta = 15$$

من الواضح أن هذا النموذج يحتوي على مشاكل، إذا كان النموذج صحيح، فإن جميع الزبائن في بياناتنا قد أعطوا إكرامية بقيمة 15%. ولكن، هذا النموذج سيعطي تخميناً معقولاً في كثير من الأحيان. بالأصح، هذا النموذج يعتبر الأكثر فائدة بالنسبة لنا إذا لم يكن لدينا أي معلومات عن النادل والزبائن سوى أنهم في الولايات المتحدة.

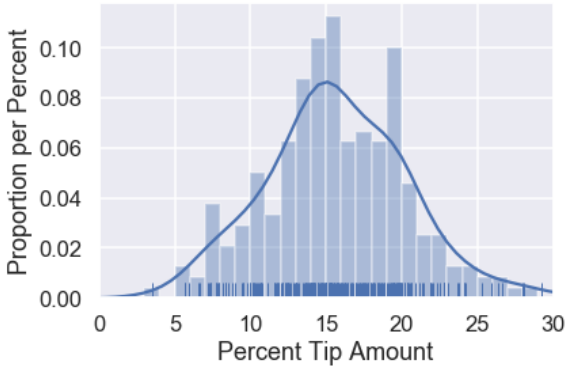
بما أن النادل لدينا قام بجمع بيانات، يمكننا استخدام المعلومات التاريخ للإكراميات التي حصل عليها لبناء نموذج يعطي تنبؤ بديل عن 15% المعروفة بين مطاعم الولايات المتحدة.

فائدة دالة الخسارة

عرضنا سابقاً توزيع نسبة الإكراميات في البيانات:

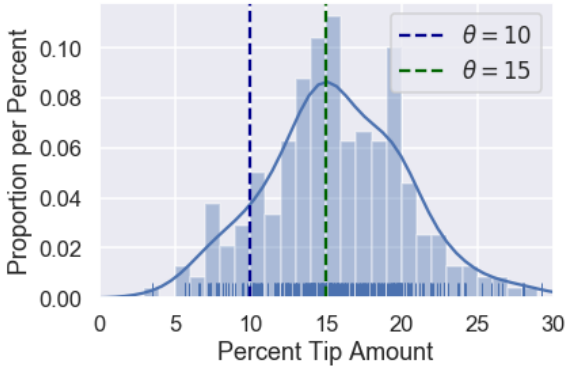
```
sns.distplot(tips['pcttip'], bins=np.arange(30), rug=True)
```

```
plt.xlim(0, 30)
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



لنفترض أننا نريد مقارنة خيارين ل θ: 10 و 15. يمكننا إظهار هذين الخيارين في رسمنا البياني:

```
sns.distplot(tips['pcttip'], bins=np.arange(30), rug=True)

plt.axvline(x=10, c='darkblue', linestyle='--', label=r'$ \theta = 10$')
plt.axvline(x=15, c='darkgreen', linestyle='--', label=r'$ \theta = 15$')
plt.legend()

plt.xlim(0, 30)
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



بشكل واضح، يظهر أن اختيار $\theta = 15$ يبدو أفضل من $\theta = 10$ بِنَاءًا على البيانات. لماذا؟ عندما نلاحظ نقاط البيانات، نرى أن أكثر البيانات كانت أقرب ل 15 من 10.

على الرغم أن $\theta = 15$ تبدو خيارًا أفضل، لا نعرف ما إذا كانت هي أفضل من $\theta = 16$. لنقوم باختيار دقيق بين مختلف احتمالات θ، نريد أن نعين لكل θ رقم متغير يقيس ما إذا كان مناسب لبياناتنا. يعني ذلك، نحتاج لدالة تأخذ قيمه ل θ وقيم من بياناتنا، وتنتج لنا رقماً يمكننا استخدامه لاختيار أفضل القيم ل θ.

تسمى هذه الدالة بـ **دالة الخسارة Loss Function**.

دالة الخسارة

لنذكر فرضيتنا السابقة: افترضنا أن هناك نسبه واحدة للإكرامية بين جميع المجتمع θ^*. نموذجنا يتوقع هذا الرقم؛ سنستخدم θ لترميز توقعنا. نريد استخدام البيانات التي جمعناها عن الإكراميات لحساب قيمة θ.

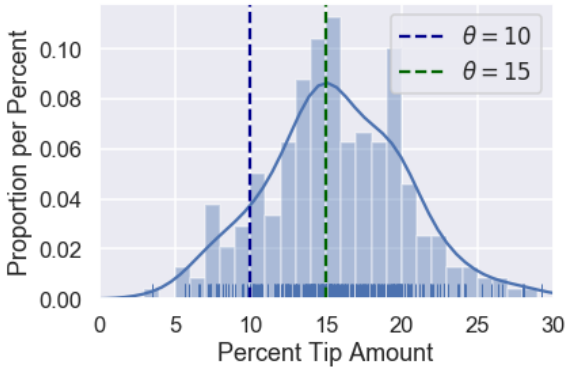
لتحديد أي قيمه من θ هي الصحيحة، عرفنا **دالة الخسارة**. دالة الخسارة هي دالة رياضية تأخذ توقع ل θ ونقطة في بياناتنا $y_1, y_2, \ldots, y_n$، وتأتي بنتيجة عبارة عن رقم، **الخسارة**، يقيس مدى دقة θ لبياناتنا. برموز رياضية، نريد كتابة الدالة:

$$L(\theta, y_1, y_2, \ldots, y_n) = \ \ldots$$

بشكل عام، النتيجة من دالة الخسارة عندما تكون رقماً صغير فإن قيمة θ مناسبة وإذا كانت النتيجة من الدالة رقم كبير فإن قيمة θ ليست مناسبه.
لندرب النموذج، نختار قيم ل θ التي تنتج لنا أرقام صغيرة في دالة الخسارة، القيم θ التي **تقلل من الخسارة**. نستخدم الرمز $\hat{\theta}$ للإشارة لقيم θ التي تقلل من دالة الخسارة.

لنفترض مرة أخرى القيم التالية ل θ: $\theta = 10$ و $\theta = 15$:

```
sns.distplot(tips['pcttip'], bins=np.arange(30), rug=True)

plt.axvline(x=10, c='darkblue', linestyle='--', label=r'$ \theta = 10$')
plt.axvline(x=15, c='darkgreen', linestyle='--', label=r'$ \theta = 15$')
plt.legend()

plt.xlim(0, 30)
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



بما أن $\theta = 15$ أقرب للكثير من النقاط، فإن دالة الخسارة من المفترض أن تأتي رقم أقل ل $\theta = 15$ ورقم أعلى ل $\theta = 10$.

لنستخدم هذه المعلومه لإنشاء دالة الخسارة.

دالة الخسارة الأولى لنا: الخطأ التربيعي المتوسط

نريد ان تكون قيمة θ قريبه من الكثير من النقاط في بياناتنا. لذا، نقوم بتعريف دالة خسارة نطبقها على بياناتنا وتأتي بنتائج عاليه ل θ إذا كانت بعيدة عن الكثير من البيانات. نبدأ بدالة خسارة بسيطه يطلع عليها *الخطأ التربيعي المتوسط Mean Squared Error*، هنا فكرتها:

- نختار قيمه ل θ.
- لكل قيمه في بياناتنا، نأخذ الفرق التربيعي بين تلك القيمة و θ: $(y_i - \theta)^2$. طريقة التربيع هي وسيلة سهله لتحويل القيم السلبية إلى قيم إيجابية. نريد القيام بذلك لأن إذا كانت $(y_i = 14)$ فإن كلا القيمتين $\theta = 10$ و $\theta = 18$ يعتبران بعيدين عن باقي النقاط ويعتبرون قيم غير مناسبة.
- للقيام بعملية الحساب لدالة الخسارة، نأخذ متوسط كل قيمه للفرق التربيعي.

$$
\begin{aligned}
L(\theta, y_1, y_2, \ldots, y_n) &= \text{average}\left\{(y_1 - \theta)^2, (y_2 - \theta)^2, \ldots, (y_n - \theta)^2\right\} \\
&= \frac{1}{n}\left((y_1 - \theta)^2 + (y_2 - \theta)^2 + \ldots + (y_n - \theta)^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2
\end{aligned}
$$

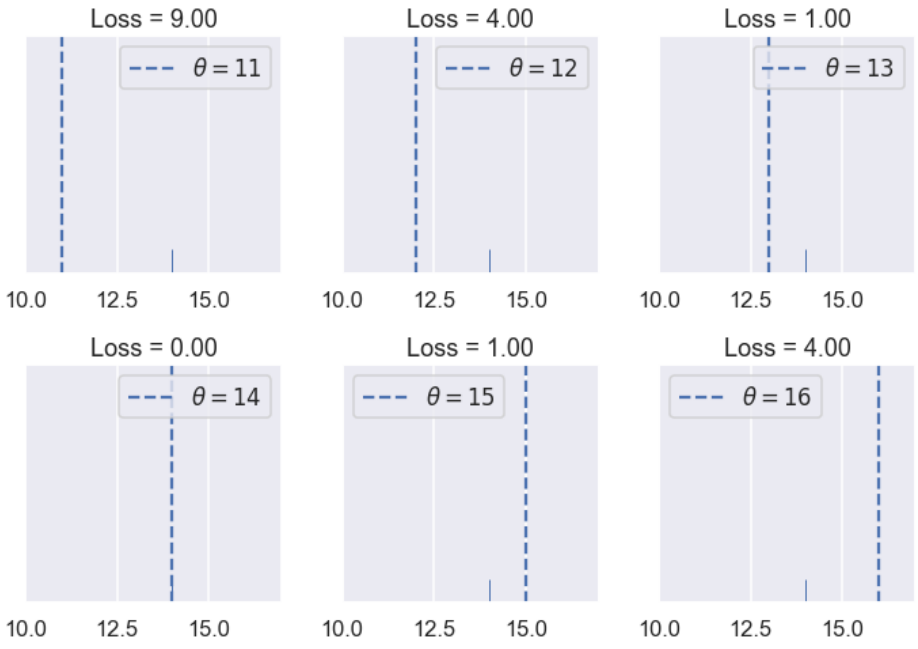
كتابة دالة في بايثون للقيام بعملية حساب دالة الخسارة هذه سهل جداً:

```
def mse_loss(theta, y_vals):
    return np.mean((y_vals - theta) ** 2)
```

لنرى كيفية أداء هذه الدالة. لنفرض أن لدينا بيانات تحتوي على قيمة واحد فقط، $y_1 = 14$. يمكننا تجربة أكثر من قيمه ل θ لنرى نتيجة دالة الخسارة لكل قيمه:

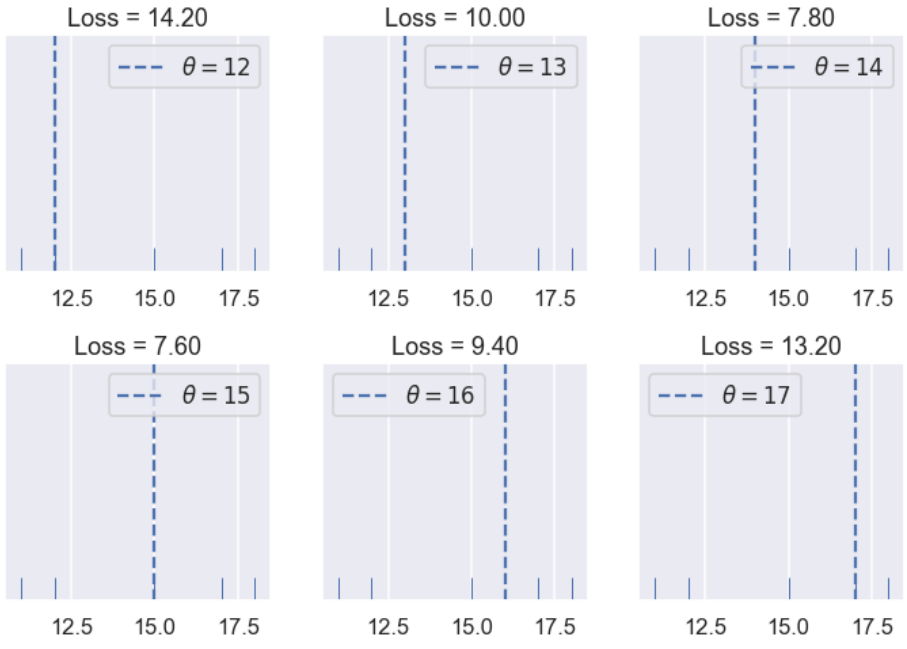
```
# لا يهم معرفة ما يجري في الكود البرمجي التالي، نتيجة الكود هي الصورة في الأسفل
def try_thetas(thetas, y_vals, xlims, loss_fn=mse_loss, figsize=(10, 7), cols=3):
    if not isinstance(y_vals, np.ndarray):
        y_vals = np.array(y_vals)
    rows = int(np.ceil(len(thetas) / cols))
    plt.figure(figsize=figsize)
    for i, theta in enumerate(thetas):
        ax = plt.subplot(rows, cols, i + 1)
        sns.rugplot(y_vals, height=0.1, ax=ax)
        plt.axvline(theta, linestyle='--',
                    label=rf'$ \theta = {theta} $')
        plt.title(f'Loss = {loss_fn(theta, y_vals):.2f}')
        plt.xlim(*xlims)
        plt.yticks([])
        plt.legend()
    plt.tight_layout()
```

```
try_thetas(thetas=[11, 12, 13, 14, 15, 16],
           y_vals=[14], xlims=(10, 17))
```



كما توقعنا، قيمة دالة الخسارة أعلى كلما كانت θ أبعد من مجمع البيانات وأقل كلما كانت أقرب أو في نفس النقطة. لنرى كيف يتصرف الخطأ التربيعي المتوسط إذا كانت لدينا أكثر من قيمه. لتكن بياناتنا هي القيم التالية: $[11, 12, 15, 17, 18]$

```
try_thetas(thetas=[12, 13, 14, 15, 16, 17],
           y_vals=[11, 12, 15, 17, 18],
           xlims=(10.5, 18.5))
```



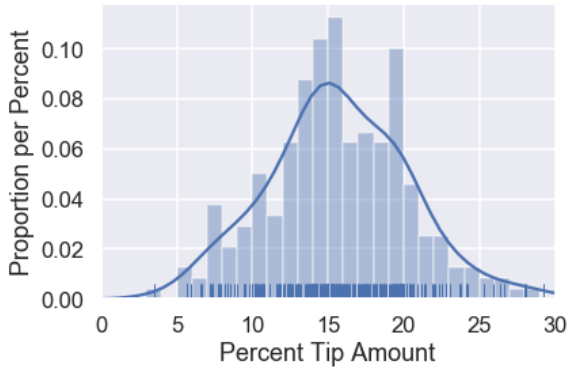
من قيم θ السابقة نرى أن $\theta = 15$ لديها أقل نتيجة لدالة الخسارة. لكن، يوجد رقم بين 14 و 15 قد يحصل على نتيجة أقل لدالة الخسارة. لنحاول البحث عن تلك القيمه الأفضل ل θ

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual
```

```
def try_thetas_interact(theta, y_vals, xlims, loss_fn=mse_loss):
    if not isinstance(y_vals, np.ndarray):
        y_vals = np.array(y_vals)
    plt.figure(figsize=(4, 3))
    sns.rugplot(y_vals, height=0.1)
    plt.axvline(theta, linestyle='--')
    plt.xlim(*xlims)
    plt.yticks([])
    print(f'Loss for theta = {theta}: {loss_fn(theta, y_vals):.2f}')

def mse_interact(theta, y_vals, xlims):
    plot = interactive(try_thetas_interact, theta=theta,
                       y_vals=fixed(y_vals), xlims=fixed(xlims),
                       loss_fn=fixed(mse_loss))
    plot.children[-1].layout.height = '240px'
    return plot

mse_interact(theta=(12, 17, 0.2),
             y_vals=[11, 12, 15, 17, 18],
             xlims=(10.5, 18.5))
```

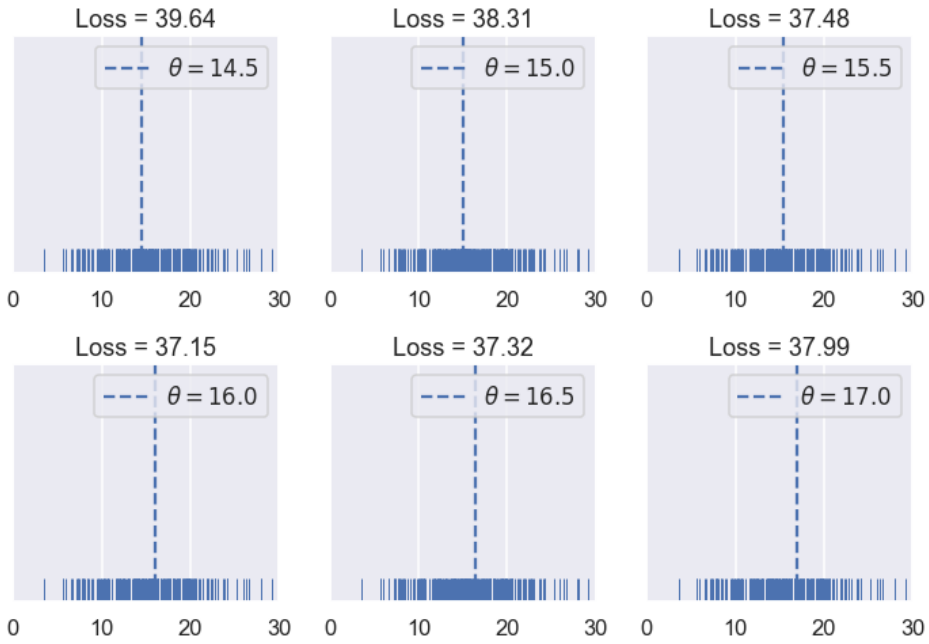
الكود السابق ينتج رسم بياني تفاعلي، يمكننا منه التحكم بقيمة θ إلى أن نحصل على نتيجة مناسبة لدالة الخسارة. هذا الطريقة التفاعلية تعمل فقط في Jupyter Notebook لذا إذا أرد تجربتها قم بتشغيل كامل الكود البرمجي مع إضافة المكتبات التي يحتاجها كاملاً في Jupyter Notebook.

نلاحظ ان دالة الخطأ التربيعي المتوسط تؤدي عملها بشكل متقن وذلك بقيامها بمعاقبة قيم θ التي تكون بعيدة عن مجموع البيانات. لنرى الآن قيمة دالة الخسارة لبيانات النسبة المئوية للإكرامية، نرسم التوزيع أولا:



لنجرب الآن قيم مختلفة ل θ:

```
try_thetas(thetas=np.arange(14.5, 17.1, 0.5),
           y_vals=tips['pcttip'],
           xlims=(0, 30))
```



بنفس الطريقة السابقة، يمكن القيام برسم بياني تفاعلي لملاحظة التغير مع تغيير قيمة θ:

```
mse_interact(theta=(13, 17, 0.25),
             y_vals=tips['pcttip'],
             xlims=(0, 30))
```

نلاحظ ان افضل قيمة ل θ هي 16.00، اعلى بقليل من توقعنا الأول 15% لكل إكرامية.

تبسيط الدالة

قمنا بتعريف أول دالة خسارة لنا، الخطأ التربيعي المتوسط (MSE). ينتج لنا رقم عالي لقيم θ البعيدة عن مجموع البيانات. رياضياً، يمكننا تعريف الدالة كالتالي:

$$L(\theta, y_1, y_2, \ldots, y_n) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

ستقوم الدالة بإنتاج أرقام مختلفة للخسارة عندما نقوم بأي تغير سواء ل θ أو $y_1, y_2, \ldots, y_n$. لاحظنا ذلك عند تجربتنا قيم مختلفة ل θ وعند إضافتنا قيم اضافيه للبيانات.

كاختصار، يمكننا تعريف المصفوفة: $\mathbf{y} = [y_1, y_2, \ldots, y_n]$. ثم نكتب دالة الخطأ التربيعي المتوسط كالتالي:

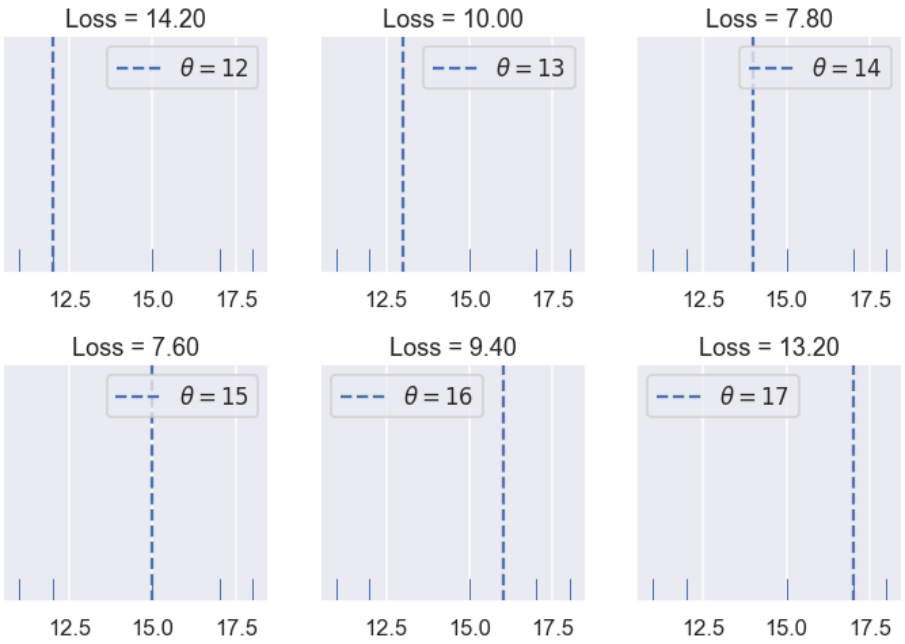
$$L(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

تقليل الخسارة

حتى الآن، وجدنا أفضل قيمه ل θ بتجربة أكثر من قيمه واختيار أقل قيمه بعد تطبيق دالة الخسارة. على الرغم من فاعلية هذه الطريقة، يمكننا إيجاد طرق أفضل باستخدام خصائص دالة الخسارة.

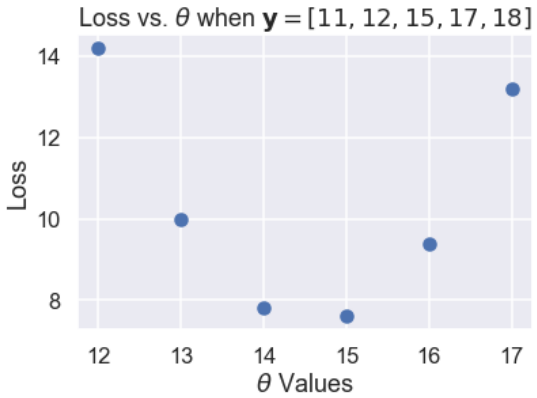
للمثال التالي، استخدمنا بيانات تحتوي على القيم التالية: $\mathbf{y} = [11, 12, 15, 16, 17]$.

```
try_thetas(thetas=[12, 13, 14, 15, 16, 17],
           y_vals=[11, 12, 15, 17, 18],
           xlims=(10.5, 18.5))
```



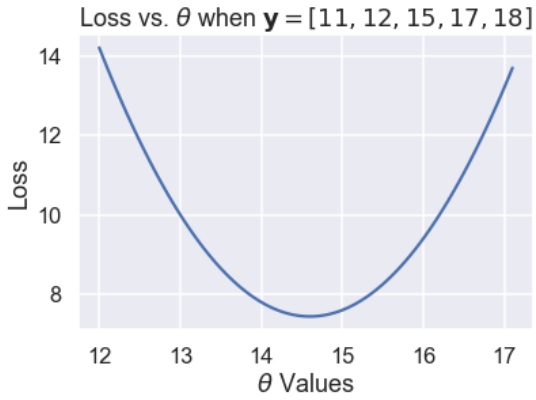
في المثال استخدمنا قيمة ل θ بين 12 و 17، عند تغير قيمة θ، تبدأ نتيجة الدالة مرتفعه (10.92) وتقل حتى نصل إلى $\theta = 15$، ثم تزيد بعدها.
يمكننا ملاحظة تغير نتيجة دالة الخسارة كلما غيرنا قيمة θ، لنقارن التغير في كل مرة نقوم فيها بتغير قيمة θ:

```
thetas = np.array([12, 13, 14, 15, 16, 17])
y_vals = np.array([11, 12, 15, 17, 18])

losses = [mse_loss(theta, y_vals) for theta in thetas]

plt.scatter(thetas, losses)
plt.title(r'Loss vs. $ \theta $ when $\bf{y}$$ = [11, 12, 15, 17, 18] $')
plt.xlabel(r'$ \theta $ Values')
plt.ylabel('Loss');
```



مخطط اَلنَّشَتُّت السابق يظهر توجه للأسفل، ثم للأعلى كما لاحظنا مسبقاً. يمكننا استخدام المزيد من قيم θ لمشاهدة كامل المنحنى عندما تتغير الخسارة كلما غيرنا قيمة θ.

```
thetas = np.arange(12, 17.1, 0.05)
y_vals = np.array([11, 12, 15, 17, 18])
losses = [mse_loss(theta, y_vals) for theta in thetas]

plt.plot(thetas, losses)
plt.title(r'Loss vs. $ \theta $ when $\bf{y}$$ = [11, 12, 15, 17, 18] $')
plt.xlabel(r'$ \theta $ Values')
plt.ylabel('Loss');
```



الرسم البياني السابق يؤكد لنا أن $\theta = 15$ لم تكن الخيار الأفضل؛ رقم ل θ بين 14 و 15 كان قيمة الخسارة فيه أقل. يمكننا استخدام التفاضل والتكامل وحساب قيمة ذلك الرقم بالضبط. بقيمة خسارة أقل، مشتقة دالة الخسارة للقيمة θ تكون 0.

نبدأ أولاً بدالة الخسارة التي عرفناها:

$$L(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

ثم نقوم بإدخال نقاطنا $\mathbf{y} = [11, 12, 15, 17, 18]$ للمعادلة:

$$L(\theta, \mathbf{y}) = \frac{1}{5}\big((11-\theta)^2 + (12-\theta)^2 + (15-\theta)^2 + (17-\theta)^2 + (18-\theta)^2\big)$$

لحساب قيمة θ التي تقلل من دالة الخسارة، نقوم بحساب المشتقة ل θ:

$$\begin{aligned}\frac{\partial}{\partial\theta}L(\theta, \mathbf{y}) &= \frac{1}{5}\big(-2(11-\theta) - 2(12-\theta) - 2(15-\theta) - 2(17-\theta) - 2(18-\theta)\big) \\ &= \frac{1}{5}\big(10\cdot\theta - 146\big)\end{aligned}$$

ثم نقوم بالبحث عن قيمة θ التي مشتقتُها تساوي صفر:

$$\begin{aligned}\frac{1}{5}\big(10\cdot\theta - 146\big) &= 0 \\ 10\cdot\theta - 146 &= 0 \\ \theta &= 14.6\end{aligned}$$

وجدنا قيمة θ الأقل، وكما هو متوقع، كانت بين 14 و 15. نرمز لقيمة θ ذات الأقل خسارة بالرمز $\hat{\theta}$. الآن، للبيانات التالية $\mathbf{y} = [11, 12, 15, 17, 18]$ و دالة الخسارة MSE:

$$\hat{\theta} = 14.6$$

إذا حسبنا متوسط البيانات لدينا، نلاحظ التالي:

$$\text{mean}(\mathbf{y}) = \hat{\theta} = 14.6$$

تقليل الخطأ التربيعي المتوسط

كما هو واضح، النتيجة السابقة (تساوي قيمة المتوسط مع $\hat{\theta}$) ليست مجرد صدفه؛ المتوسط لقيم البيانات *دائماً* ما يساوي $\hat{\theta}$، وقيمة θ التي تقلل من الخسارة بدالة MSE.

لعرض ذلك، نأخذ مشتقة دالة الخسارة مره أخرى. ولكن بدلاً من إضافة البيانات إليها، نبقي الرمز y_i لنعممها على جميع البيانات.

$$\begin{aligned}L(\theta, \mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2 \\ \frac{\partial}{\partial\theta}L(\theta, \mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n}-2(y_i - \theta) \\ &= -\frac{2}{n}\sum_{i=1}^{n}(y_i - \theta)\end{aligned}$$

بما أننا لم نعوض أي قيمة للمتغير y_i، يمكن أن نستخدم المعادلة على أي بيانات بأي حجم.

الآن، نعوض قيمة المشتقة بصفر ونحاول إيجاد θ لإيجاد أقل قيمة لدالة الخسارة كما فعلنا سابقاً:

$$
\begin{aligned}
-\frac{2}{n}\sum_{i=1}^{n}(y_i-\theta) &= 0 \\
\sum_{i=1}^{n}(y_i-\theta) &= 0 \\
\sum_{i=1}^{n}y_i - \sum_{i=1}^{n}\theta &= 0 \\
\sum_{i=1}^{n}\theta &= \sum_{i=1}^{n}y_i \\
n\cdot\theta &= y_1+\ldots+y_n \\
\theta &= \frac{y_1+\ldots+y_n}{n} \\
\hat{\theta} = \theta &= \text{mean}(\mathbf{y})
\end{aligned}
$$

نلاحظ كما هو واضح، أن هناك قيمه واحدة ل θ التي تعطي أقل قيمه ل MSE أياً كانت القيم داخل البيانات. للخطأ التربيعي المتوسط، نعلم أن $\hat{\theta}$ هي دائماً تساوي المتوسط للبيانات.

العودة للبيانات الأصلية

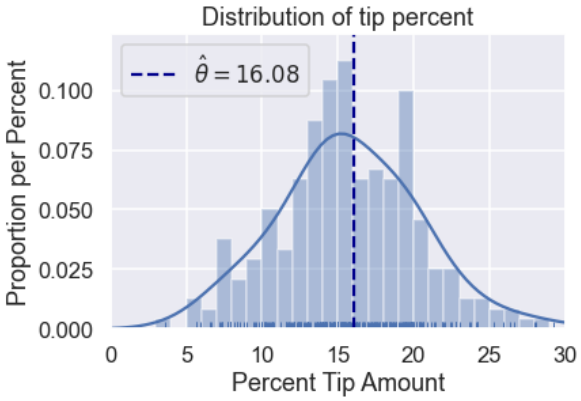
ليس علينا اختبار أي قيم أعلى ل θ كما فعلنا مسبقاً. يمكننا إيجاد متوسط الإكراميات ويكفي:

```
np.mean(tips['pcttip'])
```

```
16.08025817225047
```

لنرسمه:

```
sns.distplot(tips['pcttip'], bins=np.arange(30), rug=True)

plt.axvline(x=16.08, c='darkblue', linestyle='--', label=r'$ \hat \theta = 16.08$')
plt.legend()

plt.xlim(0, 30)
plt.title('Distribution of tip percent')
plt.xlabel('Percent Tip Amount')
plt.ylabel('Proportion per Percent');
```



ملخص دالة الخسارة

قمنا بالتعرف على **نموذج ثابت**، نموذج دائمًا ما ينتج لنا نفس الرقم لكل بياناتنا.

و تعرفنا على **دالة الخسارة** $L(\theta, \mathbf{y})$ التي تقيس مدى كفاءة قيمه معينة ل θ للبيانات. في هذا الجزء تعرفنا أيضا على دالة خسارة الخطأ التربيعي المتوسط وأظهرنا أن $\hat{\theta} = \text{mean}(\mathbf{y})$ للنموذج الثابت.

الطريقة التي استخدمناها في هذا الجزء والتي تطبق على اي نوع من النماذج:

- اختيار النموذج.
- اختيار دالة الخسارة.

- تدريب النموذج عن طريق تقليل قيمة الخسارة.

في هذا الكتاب، جميع طرق إنشاء النماذج تبدأ وتتوسع من أي من هذه النقاط.

دالتي الخسارة Absolute و Huber

لتدريب نموذج، نختار دالة خسارة ومتغيرات النموذج التي تقلل من الخسارة. في الجزء السابق، تعرفنا على دالة الخسارة MSE:

$$L(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

استخدمنا نموذج ثابت يتوقع نفس الرقم θ لجميع البيانات. عندما دربنا النموذج باستخدام دالة الخسارة MSE، وجدنا أن $\hat{\theta} = \text{mean}(\mathbf{y})$. في بيانات الإكراميات، وجدنا أن النموذج يتوقع 16.8% كونها هي متوسط نسب الإكراميات.

في هذا الجزء، سنتعرف على دالتين خسارة هما **متوسط الخطأ الحتمي (Mean Absolute Error (MAE** و **هوبر Huber**.

متوسط الخطأ الحتمي

الآن، سنبقي النموذج كما هو لكن سنغير دالة الخسارة. بدلاً من أخذ الفرق التربيعي بين كل نقطة وتوقعنا، هذه الدالة تأخذ الفرق الحتمي:

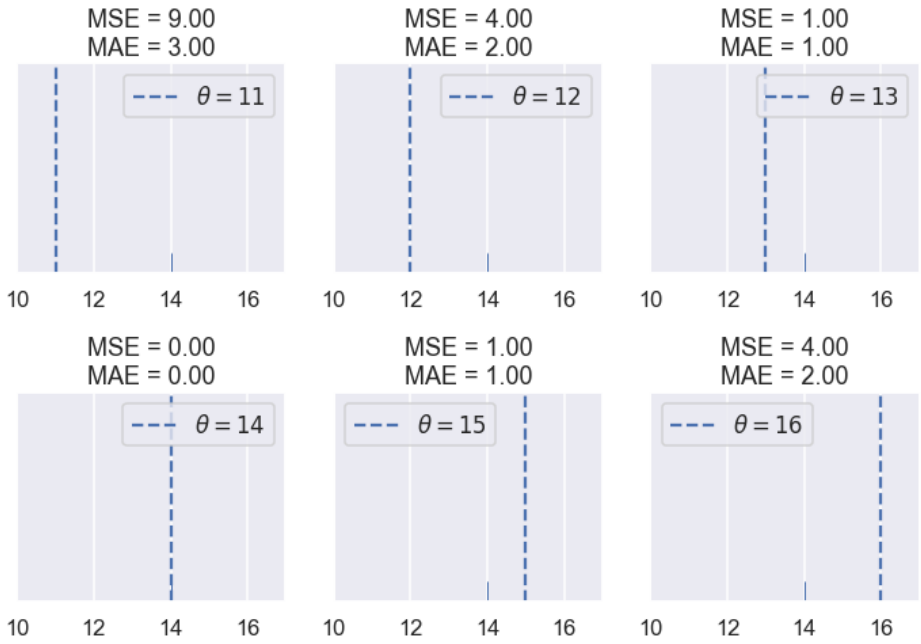
$$L(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \theta|$$

مقارنة بين MSE و MAE

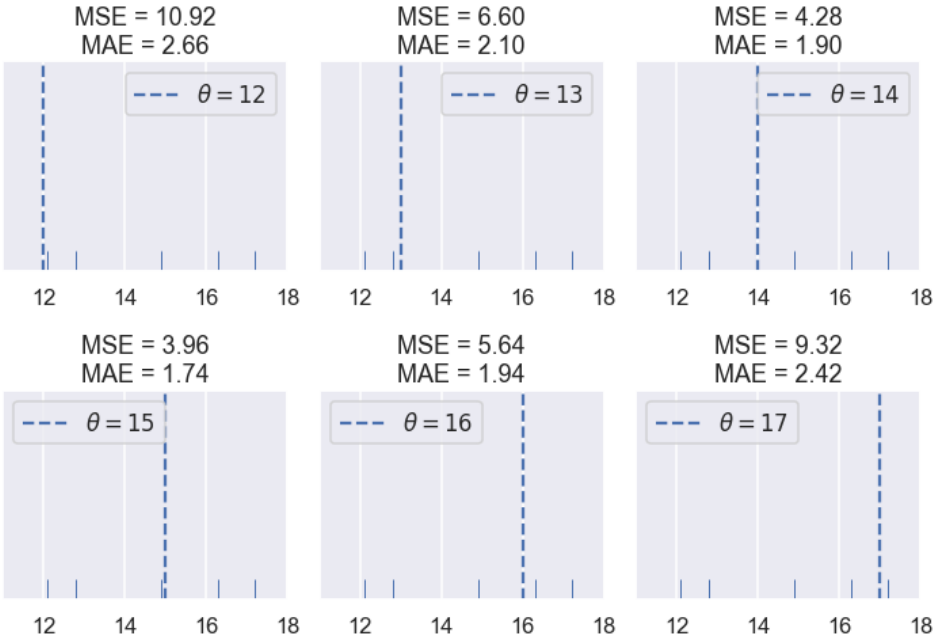
يعرف الكاتب أولاً دوال سيستخدمها للمقارنة بين MSE و MAE. معرفة معنى الكود البرمجي هنا لا يهم.

```
tips = sns.load_dataset('tips')
tips['pcttip'] = tips['tip'] / tips['total_bill'] * 100

def mse_loss(theta, y_vals):
    return np.mean((y_vals - theta) ** 2)

def abs_loss(theta, y_vals):
    return np.mean(np.abs(y_vals - theta))


def compare_mse_abs(thetas, y_vals, xlims, figsize=(10, 7), cols=3):
    if not isinstance(y_vals, np.ndarray):
        y_vals = np.array(y_vals)
    rows = int(np.ceil(len(thetas) / cols))
    plt.figure(figsize=figsize)
    for i, theta in enumerate(thetas):
        ax = plt.subplot(rows, cols, i + 1)
        sns.rugplot(y_vals, height=0.1, ax=ax)
        plt.axvline(theta, linestyle='--',
                    label=rf'$ \theta = {theta} $')
        plt.title(f'MSE = {mse_loss(theta, y_vals):.2f}\n'
                  f'MAE = {abs_loss(theta, y_vals):.2f}')
        plt.xlim(*xlims)
        plt.yticks([])
        plt.legend()
    plt.tight_layout()
```

```
compare_mse_abs(thetas=[11, 12, 13, 14, 15, 16],
                y_vals=[14], xlims=(10, 17))
```



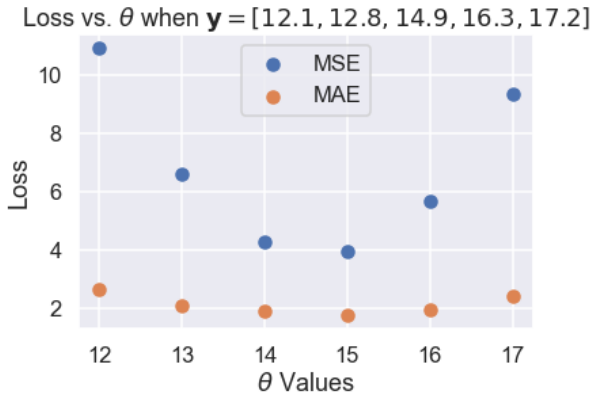
نلاحظ أن MSE عادة ما تكون أعلى من MAE كون دالة الخسارة فيها تربيعية. لنرى ما يحدث عندما يكون لدينا الخمس نقاط التالية:
$\mathbf{y} = [12.1, 12.8, 14.9, 16.3, 17.2]$



تذكر أن نتيجة دالة الخسارة لا تهمنا كثيراً؛ فهي تهمنا فقط بمقارنة القيمة المختلفة ل θ. عندما نختار دالة خسارة، سنبحث عن $\hat{\theta}$، قيمة θ التي لديها أقل قيمه للخسارة. لذا، نحن نهتم ما ذا كانت دالة الخسارة تنتج نتائج مختلفة ل $\hat{\theta}$.

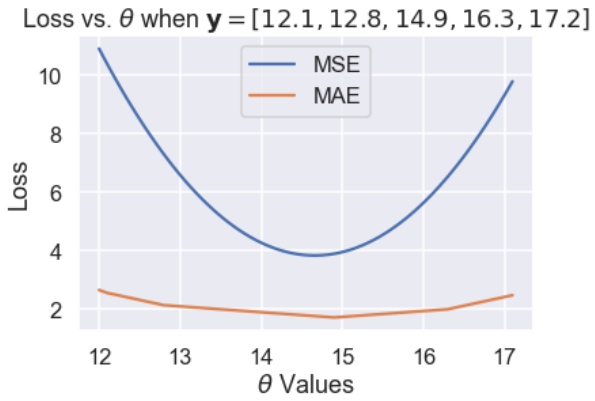
حتى الآن، كلا دالتي الخسارة متفقتان على $\hat{\theta}$. لو لاحظنا، سنرى بعض الاختلافات. لنرسم نتائج الخسارة مع θ لكل قيمه لدينا:

```
thetas = np.array([12, 13, 14, 15, 16, 17])
y_vals = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
mse_losses = [mse_loss(theta, y_vals) for theta in thetas]
abs_losses = [abs_loss(theta, y_vals) for theta in thetas]

plt.scatter(thetas, mse_losses, label='MSE')
plt.scatter(thetas, abs_losses, label='MAE')
plt.title(r'Loss vs. $ \theta $ when $ \bf{y}$$= [ 12.1, 12.8, 14.9, 16.3, 17.2 ] $')
plt.xlabel(r'$ \theta $ Values')
plt.ylabel('Loss')
plt.legend();
```



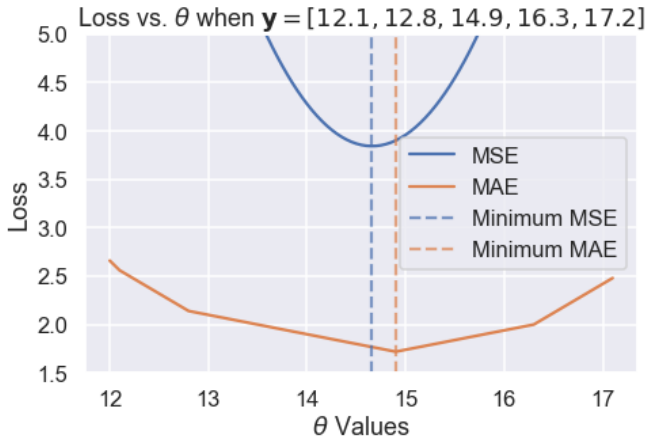
ثم، نستخدم قيم أكثر ل θ ليكون المنحنى أكثر سلاسة:

```
thetas = np.arange(12, 17.1, 0.05)
y_vals = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
mse_losses = [mse_loss(theta, y_vals) for theta in thetas]
abs_losses = [abs_loss(theta, y_vals) for theta in thetas]

plt.plot(thetas, mse_losses, label='MSE')
plt.plot(thetas, abs_losses, label='MAE')
plt.title(r'Loss vs. $ \theta $ when $ \bf{y}$$ = [ 12.1, 12.8, 14.9, 16.3, 17.2 ] $')
plt.xlabel(r'$ \theta $ Values')
plt.ylabel('Loss')
plt.legend();
```



ثم، نقرب الرسم البياني إلى القيم بين 1.5 و 5 على المحور y لنرى الفرق على القيم الدنيا بشكل أوضح. حدثنا القيم الدنيا بخطوط مُنَقَّطَة:

```
thetas = np.arange(12, 17.1, 0.05)
y_vals = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
mse_losses = [mse_loss(theta, y_vals) for theta in thetas]
abs_losses = [abs_loss(theta, y_vals) for theta in thetas]

plt.figure(figsize=(7, 5))
plt.plot(thetas, mse_losses, label='MSE')
plt.plot(thetas, abs_losses, label='MAE')
plt.axvline(np.mean(y_vals), c=sns.color_palette()[0], linestyle='--',
            alpha=0.7, label='Minimum MSE')
plt.axvline(np.median(y_vals), c=sns.color_palette()[1], linestyle='--',
            alpha=0.7, label='Minimum MAE')


plt.title(r'Loss vs. $ \theta $ when $ \bf{y}$$ = [ 12.1, 12.8, 14.9, 16.3, 17.2 ] $')
plt.xlabel(r'$ \theta $ Values')
plt.ylabel('Loss')
plt.ylim(1.5, 5)
plt.legend()
plt.tight_layout();
```



وجدنا بعد تجربتنا أن MSE و MAE قد ينتج عنهما قيم مختلفة ل $\hat{\theta}$ لنفس البيانات. تحليل أكثر دقه أوضح لنا متى يكون الاختلاف، وسببه.

القيم الشاذة

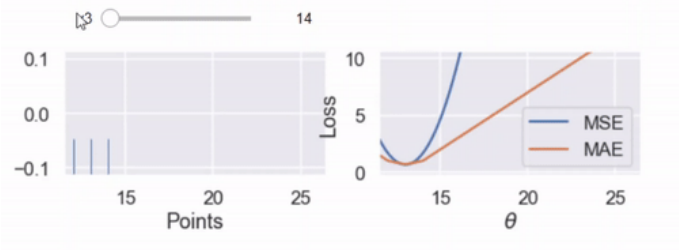
أحد الاختلافات التي نلاحظها في الرسم البياني السابق لقيمة الخسارة و θ هو شكل منحنى الخسارة. رسم MSE أنتج لنا منحنى قطعي مُكافئ.

بينما رسم MAE أنتج لنا لما يبدو وكأنها خطوط متصلة ببعضها. هذا يبدو مفهوماً إذا ما أخذنا بالاعتبار أن دالة القيمة الحتمية هي خطيه، لذا أخذ المتوسط للكثير من الدوال الحتمية سيظهر لنا نواتج شبه خطيه.

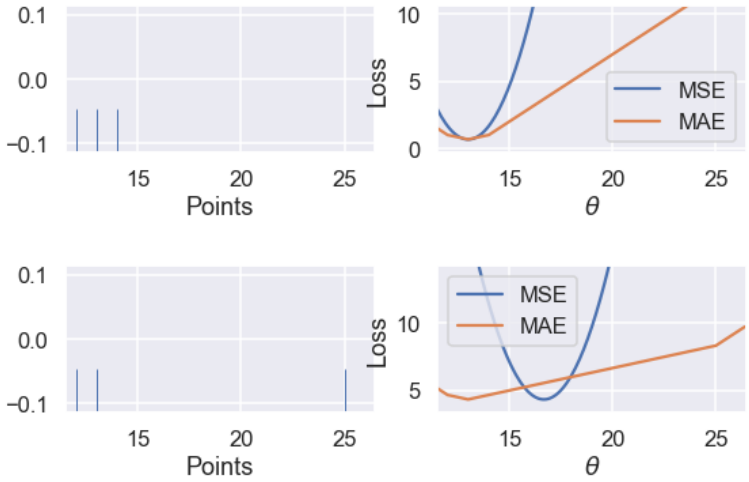
بما أن MSE تعتمد على الخطأ التربيعي، ستتأثر أكثر بالقيم الشاذة. إذا كانت $\theta = 10$ واحد النقاط كانت 110، قيمة الخطأ في MSE لهذه القيمة ستكون $(10 - 110)^2 = 10000$ بينما في MAE، ستكون $|10 - 110| = 100$. يمكننا رسم ذلك عن طريق أخذ مصفوفة من ثلاث نقاط $\mathbf{y} = [12, 13, 14]$ والمقارنة بين منحنيا الخسارة و θ ل MSE و MAE.

```
# دالة تم كتابتها لتساعد على ايجاد النتائج ورسم المقارنة
# لا تحتاج للشرح
def compare_mse_abs_curves(y3=14):
    thetas = np.arange(11.5, 26.5, 0.1)
    y_vals = np.array([12, 13, y3])

    mse_losses = [mse_loss(theta, y_vals) for theta in thetas]
    abs_losses = [abs_loss(theta, y_vals) for theta in thetas]
    mse_abs_diff = min(mse_losses) - min(abs_losses)
    mse_losses = [loss - mse_abs_diff for loss in mse_losses]

    plt.figure(figsize=(9, 2))

    ax = plt.subplot(121)
    sns.rugplot(y_vals, height=0.3, ax=ax)
    plt.xlim(11.5, 26.5)
    plt.xlabel('Points')

    ax = plt.subplot(122)
    plt.plot(thetas, mse_losses, label='MSE')
    plt.plot(thetas, abs_losses, label='MAE')
    plt.xlim(11.5, 26.5)
    plt.ylim(min(abs_losses) - 1, min(abs_losses) + 10)
    plt.xlabel(r'$ \theta $')
    plt.ylabel('Loss')
    plt.legend()


interact(compare_mse_abs_curves, y3=(14, 25));
```

كما شرحت سابقاً على الدوال التفاعلية التي قام بتعريفها الكاتب، هذه مثال آخر لها حيث يتحكم الرسم التفاعلي بقيمة المتغير الأخير y_3 نلاحظ تغير الرسم كلما كبرت قيمة المتغير من 14 إلى 25.



نلاحظ التغيير في المنحنى عندما تكون $y_3 = 14$ و $y_3 = 25$



كلما أبعدنا النقطة y_3 عن بقية البيانات، يتحرك منحنى MSE معه. عندما تكون $y_3 = 14$ فإن $\hat{\theta} = 13$ في MSE و MAE. ولكن عندما تكون $y_3 = 25$ فإن خسارة MSE تكون $\hat{\theta} = 16.7$ و MAE $\hat{\theta} = 13$، دون تغيير كما كانت مُسبقاً.

تقليل متوسط الخطأ الحتمي

الآن بعد أن تعرفنا على القليل من الفروقات بين MSE و MAE، يمكننا تقليل خسارة MAE لنجعل الفرق بينهما أكثر وضوحاً. كما فعلنا مسبقاً، سنأخذ مشتقة دالة الخسارة ل θ ونجعلها تساوي صفر.

ولكن، المرة، يجب أن نتعامل مع احتمالية أن الدالة الحتمية ليست دائماً قابله للاشتقاق. عندما تكون $x > 0$ فإن $\frac{\partial}{\partial x}|x| = 1$. عندما تكون $x < 0$ فإن $\frac{\partial}{\partial x}|x| = -1$. على الرغم ان $|x|$ ليست قابله للاشتقاق عندما تكون $x = 0$، سنجعل $\frac{\partial}{\partial x}|x| = 0$ لتكون المعادلة سهله للتعامل.

لنتذكر أن معادلة MAE هي:

$$\begin{aligned} L(\theta, \mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n}|y_i - \theta| \\ &= \frac{1}{n}\left(\sum_{y_i<\theta}|y_i - \theta| + \sum_{y_i=\theta}|y_i - \theta| + \sum_{y_i>\theta}|y_i - \theta|\right) \end{aligned}$$

في الجزء الأخير فصلنا الجمع إلى ثلاثة أشكال: واحد عندما تكون $y_i < \theta$ و أخرى عند $y_i = \theta$ وأخيره عندما يكون لدينا $y_i > \theta$. لماذا نجعل عملية الجمع معقدة بهذا الشكل؟ إذا علمنا أن $y_i < \theta$ فإننا نعلم أن $|y_i - \theta| < 0$ ولذلك فإن $\frac{\partial}{\partial \theta}|y_i - \theta| = -1$ كما في السابق. وينطبق نفس المنطق على جميع المصطلحات السابق ذكرها ليكون أخذ المشتقات أكثر سهوله.

الآن، نأخذ المشتقة بالنسبة ل θ ونجعلها تساوي صفر:

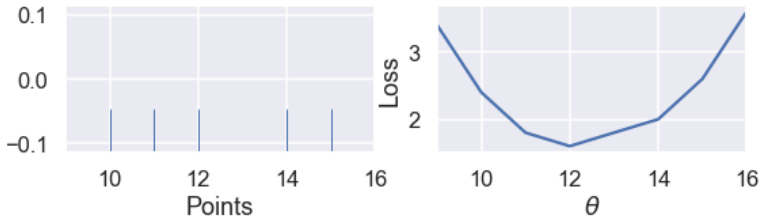
$$\begin{aligned} \frac{1}{n}\left(\sum_{y_i<\theta}(-1) + \sum_{y_i=\theta}(0) + \sum_{y_i>\theta}(1)\right) &= 0 \\ \sum_{y_i<\theta}(-1) + \sum_{y_i>\theta}(1) &= 0 \\ -\sum_{y_i<\theta}(1) + \sum_{y_i>\theta}(1) &= 0 \\ \sum_{y_i<\theta}(1) &= \sum_{y_i>\theta}(1) \end{aligned}$$

ماذا تعني النتيجة في الأعلى؟ على اليسار، لدينا قيمه واحدة لكل نقطة أقل من θ. وعلى اليمين، لدينا قيمه واحدة أيضاً لكل نقطة أعلى من θ. الآن، لتكون المعادلة صحيحه، نحتاج أن نختار قيمه ل θ لديها نفس الرقم لنقطة أقل وأعلى. هذا هو تعريف *الوسيط Median* لمجموعه من الأرقام. لذا، القيمة الأقل ل θ في MAE هي $\hat{\theta} = \text{median}(\mathbf{y})$.

عندما يكون لدينا عدد فردي من النقاط، الوسيط هو ببساطه النقطة في الوسط بعد ترتيب النقاط. يمكن أن نرى مثال على ذاك في خمس نقاط، يتم تقليل الخسارة عندما تكون θ في الوسيط:

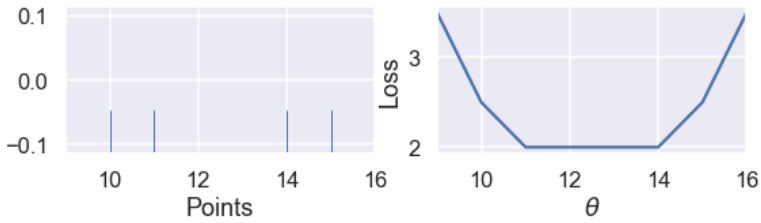
```
# مره اخرى، الكاتب عرف دالة تساعده على الحساب والرسم
# لا تحتاج لمعرفه معنى الكود البرمجي
def points_and_loss(y_vals, xlim, loss_fn=abs_loss):
    thetas = np.arange(xlim[0], xlim[1] + 0.01, 0.05)
    abs_losses = [loss_fn(theta, y_vals) for theta in thetas]

    plt.figure(figsize=(9, 2))
```

```
    ax = plt.subplot(121)
    sns.rugplot(y_vals, height=0.3, ax=ax)
    plt.xlim(*xlim)
    plt.xlabel('Points')

    ax = plt.subplot(122)
    plt.plot(thetas, abs_losses)
    plt.xlim(*xlim)
    plt.xlabel(r'$ \theta $')
    plt.ylabel('Loss')

points_and_loss(np.array([10, 11, 12, 14, 15]), (9, 16))
```



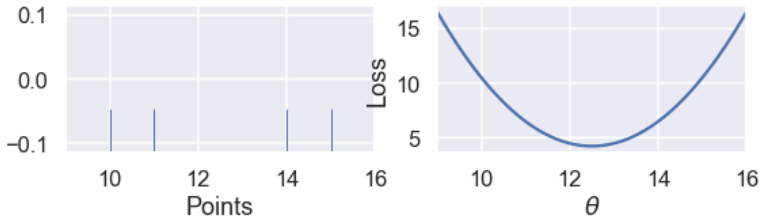
ولكن، عندما يكون لدينا عدد زوجي من الأرقام، تُقَلّل الخسارة عندما تكون θ هي القيمة التي بين النقطتين في الوسط. مثال:

```
points_and_loss(np.array([10, 11, 14, 15]), (9, 16))
```



ولا يطبق ذلك عندما نستخدم MSE:

```
points_and_loss(np.array([10, 11, 14, 15]), (9, 16), mse_loss)
```



مقارنة MSE و MAE

تحليلنا والمشتقات في الأعلى تُظهر لنا MSE أسهل للاستخدام وإيجاد الاشتقاق ولكنا تتأثر شكل كبير من البيانات الشاذة على عكس MAE. بالنسبة ل MSE، فإن $\hat{\theta} = \text{mean}(\mathbf{y})$، ول MAE فإن $\hat{\theta} = \text{median}(\mathbf{y})$. لاحظ أن الوسيط أقل تأثراً بالقيم الشاذة عن المتوسط. ظهر لنا ذلك أثناء تعريف وبناء دالتي الخسارة MSE و MAE.

رأينا أيضاً أن MSE لديها $\hat{\theta}$ مختلفة، فيها المتوسط الحتمي يمكن أن يكون أكثر احتمال من $\hat{\theta}$ عندما يكون عدد النقاط في البيانات زوجي.

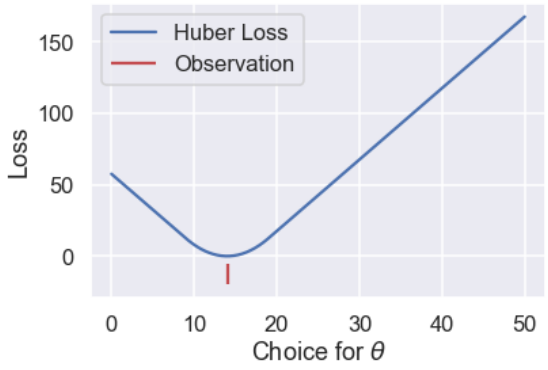
دالة الخسارة Huber

دالة الخسارة الثالثة يطلق عليها دالة Huber والتي تجمع بين MSE و MAE لتكون لنا دالة خسارة قابله للاشتقاق ولا تتأثر بالقيم الشاذة. تفعل ذلك دالة Huber عن طريق العمل وكأنها دالة MSE لقيم θ عندما تكون قريبه من القيم الدنيا والتغيير إلى MAE عندما تكون قيم θ بعيدة عن القيم الدنيا.

كما في الغالب، ننشأ دالة خسارة عن طريق أخذ المتوسط لكل خسارة Huber لكل نقطة في بياناتنا.

لنرى نتيجة دالة الخسارة Hurber عندما تكون بياناتنا عبارة عن $\mathbf{y} = [14]$ مع قيم مختلفة ل θ:

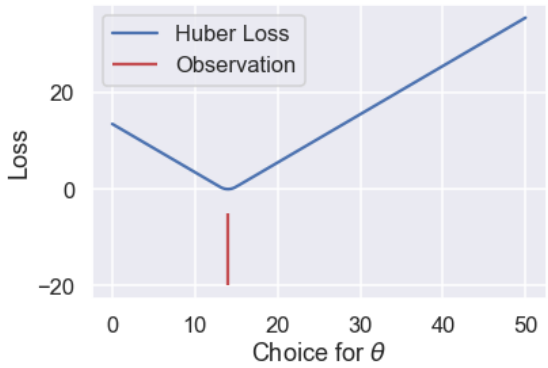
```
def huber_loss(est, y_obs, alpha = 1):
    d = np.abs(est - y_obs)
    return np.where(d < alpha,
                    (est - y_obs)**2 / 2.0,

                    alpha * (d - alpha / 2.0))

thetas = np.linspace(0, 50, 200)
loss = huber_loss(thetas, np.array([14]), alpha=5)
plt.plot(thetas, loss, label="Huber Loss")
plt.vlines(np.array([14]), -20, -5,colors="r", label="Observation")
plt.xlabel(r"Choice for $\theta$")
plt.ylabel(r"Loss")
plt.legend()
# حفظ النتيجة في ملف PDF
plt.savefig('huber_loss.pdf')
```



نلاحظ أن خط دالة الخسارة Huber سلس، على عكس MAE. أيضا تزداد خسارة Huber بمعدل خطي، على عكس المعدل التربيعي لمتوسط الخسارة التربيعية.

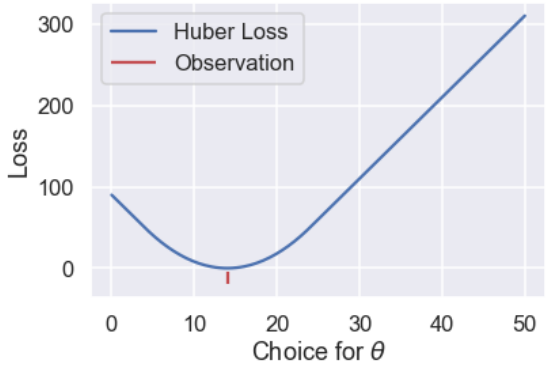
ولكن لدى دالة خسارة Huber عيب. لاحظ أنها تغيرت من MSE إلى MAE عندما كانت θ أبعد عن النقاط. يمكننا التحكم بمدى البعد للحصول على منحنيات خسارة أخرى. مثلاً، يمكننا أن نجعلها تبدأ بالتغيير عندما تكون θ أقرب بنقطة عن القيمة الدنيا:

```
loss = huber_loss(thetas, np.array([14]), alpha=1)
plt.plot(thetas, loss, label="Huber Loss")
plt.vlines(np.array([14]), -20, -5,colors="r", label="Observation")
plt.xlabel(r"Choice for $\theta$")
plt.ylabel(r"Loss")
plt.legend()
# حفظ النتيجة في ملف PDF
plt.savefig('huber_loss.pdf')
```



أو يمكننا تغير ذلك وجعلها تتغير عندما تكون θ على بعد عشر نقاط من القيمة الدنيا:

```
loss = huber_loss(thetas, np.array([14]), alpha=10)
plt.plot(thetas, loss, label="Huber Loss")
plt.vlines(np.array([14]), -20, -5,colors="r", label="Observation")
plt.xlabel(r"Choice for $\theta$")
plt.ylabel(r"Loss")
plt.legend()
# حفظ النتيجة في ملف PDF
plt.savefig('huber_loss.pdf')
```



الخيار هنا يكون لنا منحنى مختلف لخط الخسارة مما ينتج لنا أيضا قيم مختلفة ل $\hat{\theta}$. إذا أردنا استخدام دالة الخسارة Huber، لدينا مهمة أخرى وهي تحديد نقطة التغيير وجعلها قيمه مناسبة.

يمكننا تعريف دالة Huber رياضياً كالتالي:

$$L_\alpha(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} \frac{1}{2}(y_i - \theta)^2 & |y_i - \theta| \leq \alpha \\ \alpha(|y_i - \theta| - \frac{1}{2}\alpha) & \text{otherwise} \end{cases}$$

هي أكثر تعقيداً من دوال الخسارة السابقة لأنها تجمع بين MSE و MAE. المتغير الإضافي α يحدد مكان نقطة التغيير في دالة Huber من MSE إلى MAE.

محاولة إيجاد مُشتقة دالة Huber عملية مملة ولا تنتج نتائج واضحة ومفهومه كما في MSE و MAE. بدلاً من ذلك، يمكننا استخدام العملية الحسابية المسماة النزول الاشتقاقي لإيجاد القيمة الأقل ل θ.

ملخص دالتي الخسارة Absolute و Huber

في هذا الجزء، تعرفنا على دالتي خسارة: متوسط الخطأ الحتمي و دالة Huber. ووجدنا أن لنموذج ثابت باستخدام MAE فإن $\hat{\theta} = \text{median}(\mathbf{y})$.

النزول الاشتقاقي وتحسين النتائج الكمية

مقدمة

لاستخدام قاعدة بيانات للتنبؤ والتوقع، يجب علينا تكوين نموذجنا بشكل دقيق واختيار دالة خسارة. مثلاً، بيانات الإكراميات، نموذجنا توقع أن نسبة الإكرامية ثابتة لا تتغير. ثم قررنا استخدام دالة الخطأ التربيعي المتوسط MSE ووجدنا النموذج الأقل خسارة.

وجدنا أيضاً أن هناك وصفاً أبسط لدالتي الخسارة الخطأ التربيعي المتوسط و متوسط الخطأ الحتمي وهي: المتوسط والوسيط. ولكن، كلما كان نموذجنا ودالة الخسارة أكثر تعقيداً لن نستطيع إيجاد وصفاً رياضياً مناسب. مثلاً، دالة Huber لديها خصائص مفيده ولكن صعب تمييزها.

يمكننا استخدام الكمبيوتر لحل هذه المشكل بواسطة النزول الاشتقاقي، طريقه حسابية لتقليل دوال الخسارة.

تقليل الخسارة باستخدام برنامج

لنعود للنموذج من الفصل السابق:

$$\theta = C$$

سنستخدم دالة الخسارة MSE:

$$L(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2$$

للتبسيط، سنستخدم البيانات التالية: $\mathbf{y} = [12, 13, 15, 16, 17]$. نعلم من خلال تحليلنا للبيانات في الفصل السابق أن قيمة θ لدالة الخسارة MSE هي المتوسط $\text{mean}(\mathbf{y}) = 14.6$. لنرى اذا كان بإمكاننا الحصول على نتيجة عند كتابتنا لبرنامج يوجدها.

إذا قمنا بكتابة برنامج بشكل متقن، فبإمكاننا استخدام نفس البرنامج على أي دالة خسارة لإيجاد أقل قيمة ل θ، يشمل ذلك دالة Huber المعقدة رياضياً:

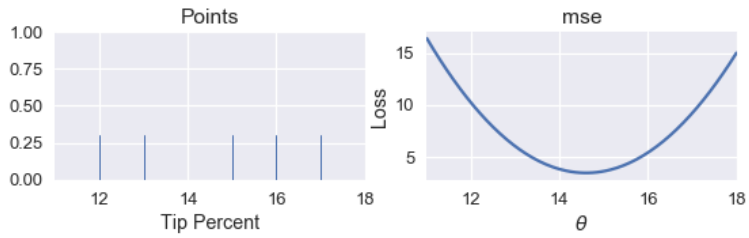
$$L_\alpha(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} \frac{1}{2}(y_i - \theta)^2 & |y_i - \theta| \leq \alpha \\ \alpha(|y_i - \theta| - \frac{1}{2}\alpha) & \text{otherwise} \end{cases}$$

أولاً، نقوم برسم تخطيطي للبيانات. بالجانب الأيمن من الرسم نقوم برسم دالة الخسارة MSE لقيم مختلفة ل θ:

```
pts = np.array([12, 13, 15, 16, 17])
points_and_loss(pts, (11, 18), mse)
```

في الكود البرمجي السابق استخدم الكاتب دالة عرفها مسبقاً بأسم points_and_loss، تقبل الدالة ثلاث متغيرات، الأولى هي البيانات. المتغير الثاني هي مقاسات أبعاد x-axis ، والقيمة الأخيرة هي نوع دالة الخسارة، والتي هي عبارة عن دالة أخرى عرفها أيضاً بأسم mse. تعريف كلا الدالتين هو كالتالي:

```
def mse(theta, y_vals):
    return np.mean((y_vals - theta) ** 2)

def points_and_loss(y_vals, xlim, loss_fn):
    thetas = np.arange(xlim[0], xlim[1] + 0.01, 0.05)
    losses = [loss_fn(theta, y_vals) for theta in thetas]

    plt.figure(figsize=(9, 2))

    ax = plt.subplot(121)
    sns.rugplot(y_vals, height=0.3, ax=ax)
    plt.xlim(*xlim)
    plt.title('Points')
    plt.xlabel('Tip Percent')

    ax = plt.subplot(122)
    plt.plot(thetas, losses)
    plt.xlim(*xlim)
    plt.title(loss_fn.__name__)
    plt.xlabel(r'$ \theta $')
    plt.ylabel('Loss')
    plt.legend()
```



كيف بإمكاننا برمجة برنامج يقوم بإيجاد أقل قيمة ل θ أوتوماتيكياً؟ الطريقة الأسهل هي بحساب قيمة الخسارة لأكثر من قيمه ل θ، ثم نوجد قيمة θ ذا الأقل خسارة.

عرفنا دالة بأسم simple_minimize والتي تقبل متغيرات هي دالة الخسارة، مصفوفة البيانات، ومصفوفة بقيم θ:

```
def simple_minimize(loss_fn, dataset, thetas):
    '''
    ذات الأقل خسارة θ من بين عدة قيم من θ القيمه النهائيه لهذه الداله هي قيمة
    '''
    losses = [loss_fn(theta, dataset) for theta in thetas]
    return thetas[np.argmin(losses)]
```

ثم نعرف دالة لإيجاد قيمة MSE واستخدامها في دالة simple_minimize:

```
def mse(theta, dataset):
    return np.mean((dataset - theta) ** 2)

dataset = np.array([12, 13, 15, 16, 17])
thetas = np.arange(12, 18, 0.1)

simple_minimize(mse, dataset, thetas)
```

```
14.599999999999991
```

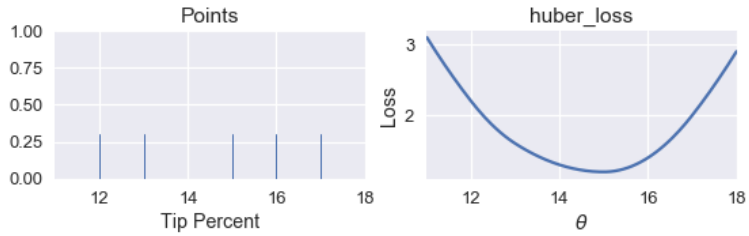
النتيجة هذه قريبه للقيمة المتوقعة:

```
# ايجاد القيمه باستخدام المتوسط
np.mean(dataset)
```

```
14.6
```

الآن يمكننا كتابة دالة لحساب خسارة Huber ورسمها:

```
def huber_loss(theta, dataset, alpha = 1):
    d = np.abs(theta - dataset)
    return np.mean(
        np.where(d < alpha,
                 (theta - dataset)**2 / 2.0,
                 alpha * (d - alpha / 2.0))
    )

points_and_loss(pts, (11, 18), huber_loss)
```



على الرغم أن القيم الدنيا ل θ يجب أن تكون أقرب إلى 15، ليس لدينا طريقه لتحليل وإيجاد قيمة θ لدالة الخسارة Huber. بدلاً، من ذلك، سنستخدم الدالة `simple_minimize`:

```
simple_minimize(huber_loss, dataset, thetas)
```

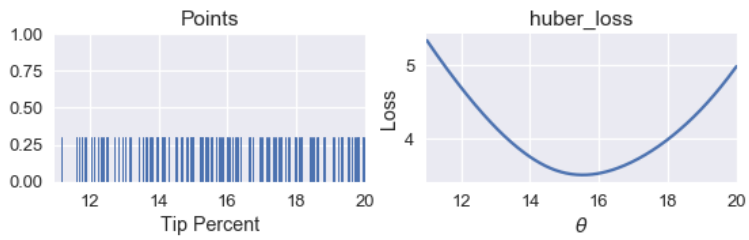
```
14.999999999999989
```

الآن، لنعود لبيانات نسبة الإكراميات ونوجد أفضل قيمه ل θ باستخدام Huber:

```
tips = sns.load_dataset('tips')
tips['pcttip'] = tips['tip'] / tips['total_bill'] * 100
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size	pcttip
0	16.99	1.01	Female	No	Sun	Dinner	2	5.944673
1	10.34	1.66	Male	No	Sun	Dinner	3	16.054159
2	21.01	3.5	Male	No	Sun	Dinner	3	16.658734
3	23.68	3.31	Male	No	Sun	Dinner	2	13.978041
4	24.59	3.61	Female	No	Sun	Dinner	4	14.680765

```
points_and_loss(tips['pcttip'], (11, 20), huber_loss)
```



```
simple_minimize(huber_loss, tips['pcttip'], thetas)
```

```
15.499999999999988
```

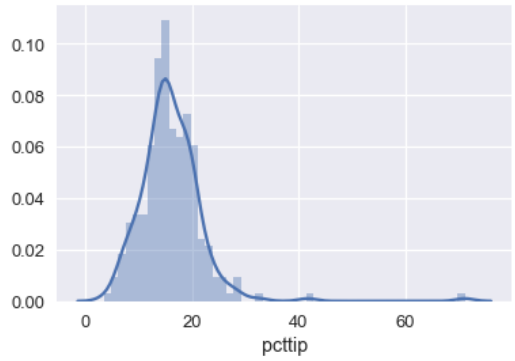
نلاحظ أن عند استخدام دالة خسارة Huber كانت النتيجة $\hat{\theta} = 15.5$. يمكننا الآن مقارنة هذه النتيجة مع MSE و MAE:

```
print(f"              MSE: theta_hat = {tips['pcttip'].mean():.2f}")
print(f"              MAE: theta_hat = {tips['pcttip'].median():.2f}")
print(f"       Huber loss: theta_hat = 15.50")
```

```
          MSE: theta_hat = 16.08
          MAE: theta_hat = 15.48
   Huber loss: theta_hat = 15.50
```

نلاحظ أن دالة Huber أقرب إلى MAE كونها لا تتأثر بشكل كبير بسبب القيم الشاذة على الجانب الأيمن في الرسم البياني التالي لتوزيع بيانات الإكراميات:

```
sns.distplot(tips['pcttip'], bins=50);
```



مشاكل simple_minimize

على الرغم من أن دالة `simple_minimize` تساعدنا على تقليل دالة الخسارة، إلا أن لديها بعض المشاكل التي تجعلها غير مفيده للاستخدام بشكل عام. مشكلتها الأهم هي أنها تعمل فقط مع قيم θ. مثلاً، في الكود البرمجي التالي، الذي سبق أن استخدمناه في الأعلى، احتجنا لتعريف قيم θ يدوياً من 12 إلى 18:

```
dataset = np.array([12, 13, 15, 16, 17])
thetas = np.arange(12, 18, 0.1)

simple_minimize(mse, dataset, thetas)
```

كيف وجدنا أنه علينا التحقق من القيم بين 12 و 18؟ احتجنا لمراجعة الرسم البياني لدالة الخسارة ووجدنا أن القيم الدنيا بين تلك القيمتين. هذه الطريقة غير عملية لأننا قمنا بإضافة خطوه معقدة جديده لنماذجنا. بالإضافة لذلك، قمنا بتعريف قيم الزيادة 0.1 بشكل يدوي. ولكن، إذا كانت القيمة المثلى ل θ هي 12.043، ستقوم الدالة `simple_minimize` بتقريب النتيجة إلى 12.00 كونها الأقرب لمضاعفات 0.1

يمكننا حل تلك المشاكل بطريقه واحدة باستخدام ما يسمى بـ *النزول الاشتقاقي Gradient Descent*.

النزول الاشتقاقي

نحن مُهتمون لبناء دالة تستطيع التقليل من دالة الخسارة دون تقييد المستخدم لتحديد قيم مسبقة ل θ للتجربة عليها. بمعنى أصح، بما أن دالة `simple_minimize` شكلها كالتالي: 📝

```
simple_minimize(loss_fn, dataset, thetas)
```

نريد دالة لديها الشكل التالي

```
minimize(loss_fn, dataset)
```

لاحظ في الشكل الذي نبحث عنه، لا يحتاج المستخدم لإضافة لقيم مسبقة ل θ في المتغيرات المطلوبة لدالة تقليل الخسارة `minimize`.

تحتاج هذه الدالة لإيجاد قيم θ الأقل خسارة أوتوماتيكياً أياً كان حجمها. سنستخدم طريقة تسمى بالنزول الاشتقاقي لبناء الدالة الجديدة المسماة `minimize`.

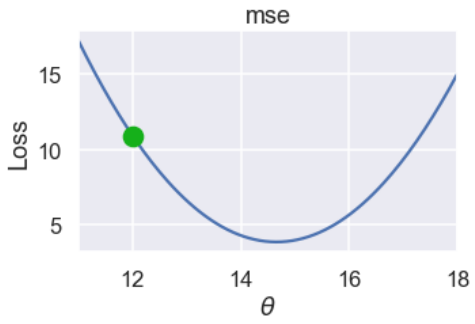
الفكرة

كما في دوال الخسارة، سنتحدث عن فكرة النزول الاشتقاقي أولاً، ثم نتعرف ونفهم العملية الرياضية فيها.

بما أن الدالة `minimize` لا يقدم لها قيم ل θ للتجربة عليها، نقوم باختيار قيمه ل θ بأي مكانٍ. ثم، نقوم بشكل تكراري بتحسين نتائج θ. وللتحسين من النتائج، نقوم بملاحظة الميلان Slope لتلك القيمة من θ التي اخترناها في الرسم البياني.

مثلاً، سنستخدم MSE على البيانات التالية $\mathbf{y} = [12.1, 12.8, 14.9, 16.3, 17.2]$ وقيمة θ التي اخترناها هي 12:

```
pts = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
plot_loss(pts, (11, 18), mse)
plot_theta_on_loss(pts, 12, mse)
```

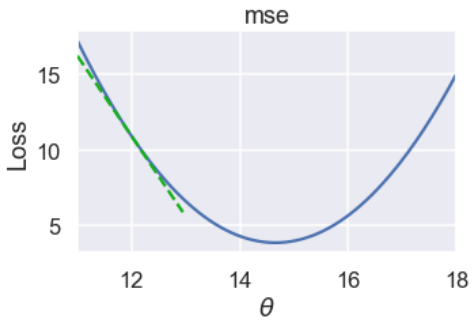


استخدم الكاتب 3 دوال كما فعل مسبقاً لتساعده على القيام بالعملية الحسابية والرسم البياني وهي `plot_theta_on_loss`، `plot_loss` و `plot_tangent_on_loss` . والكود البرمجي في الأسفل هو تعريف لكل الدوال:

```
def plot_loss(y_vals, xlim, loss_fn):
   thetas = np.arange(xlim[0], xlim[1] + 0.01, 0.05)
   losses = [loss_fn(theta, y_vals) for theta in thetas]

   plt.figure(figsize=(5, 3))
   plt.plot(thetas, losses, zorder=1)
   plt.xlim(*xlim)
   plt.title(loss_fn.__name__)
   plt.xlabel(r'$ \theta $')
   plt.ylabel('Loss')


def plot_theta_on_loss(y_vals, theta, loss_fn, **kwargs):
   loss = loss_fn(theta, y_vals)
   default_args = dict(label=r'$ \theta $', zorder=2,
                       s=200, c=sns.xkcd_rgb['green'])
   plt.scatter([theta], [loss], **{**default_args, **kwargs})


def plot_tangent_on_loss(y_vals, theta, loss_fn, eps=1e-6):
   slope = ((loss_fn(theta + eps, y_vals) - loss_fn(theta - eps, y_vals))
            / (2 * eps))
   xs = np.arange(theta - 1, theta + 1, 0.05)
   ys = loss_fn(theta, y_vals) + slope * (xs - theta)
   plt.plot(xs, ys, zorder=3, c=sns.xkcd_rgb['green'], linestyle='--')
```

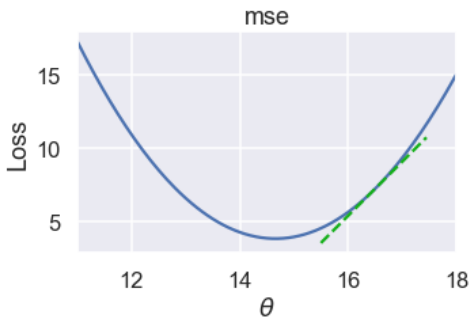
نريد اختيار قيمه جديده ل θ لتقليل الخسارة. ولعمل ذلك، كما ذكرنا سابقاً، نلاحظ الميلان لقيمة $\theta = 12$:

```
pts = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
plot_loss(pts, (11, 18), mse)
plot_tangent_on_loss(pts, 12, mse)
```



قيمة الميلان سلبية، يعني ذلك أن زيادة قيمة θ سيقلل من الخسارة. إذا كانت $\theta = 16.5$، فإن قيمة الميلان ستكون موجبه:

```
pts = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
plot_loss(pts, (11, 18), mse)
plot_tangent_on_loss(pts, 16.5, mse)
```



عندما تكون نتيجة الميلان إيجابية، فإن تقليل قيمة θ سيقلل الخسارة.

الميلان في الخط يخبرنا بأي اتجاه نختار θ لتقليل الخسارة. إذا كان الميلان نتيجته سلبية، فنحتاج لتحرك θ إلى الجانب الإيجابي. وإذا كان إيجابياً، فعلينا تحريك θ إلى الجانب السلبي. رياضاً، نقول التالي:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\partial}{\partial\theta} L(\theta^{(t)}, \mathbf{y})$$

وفيها $\theta^{(t)}$ هي القيمة الحالية، و $\theta^{(t+1)}$ هي القيمة التالية.

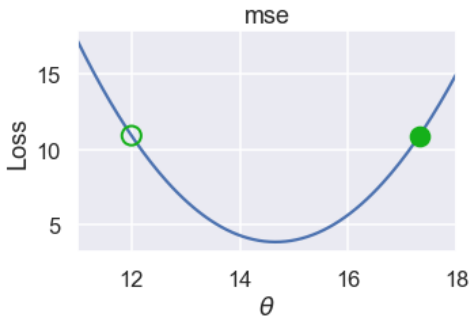
بالنسبة ل MSE، فستكون كالتالي:

$$\begin{aligned} L(\theta, \mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2 \\ \frac{\partial}{\partial\hat{\theta}} L(\theta, \mathbf{y}) &= \frac{1}{n}\sum_{i=1}^{n} -2(y_i - \theta) \\ &= -\frac{2}{n}\sum_{i=1}^{n}(y_i - \theta) \end{aligned}$$

عندما تكون $\theta^{(t)} = 12$، فالنتيجة هي $-\frac{2}{n}\sum_{i=1}^{n}(y_i - \theta) = -5.32$، ثم نستخدمها بالمعادلة السابقة:
$\theta^{(t+1)} = 12 - (-5.32) = 17.32$

رسمنا في الأسفل القيمة السابقة ل θ بدائرة مفرغة بحدود خضراء والقيمة الجديدة لها بدائرة باللون الأخضر:

```
pts = np.array([12.1, 12.8, 14.9, 16.3, 17.2])
plot_loss(pts, (11, 18), mse)
plot_theta_on_loss(pts, 12, mse, c='none',
                   edgecolor=sns.xkcd_rgb['green'], linewidth=2)
plot_theta_on_loss(pts, 17.32, mse)
```



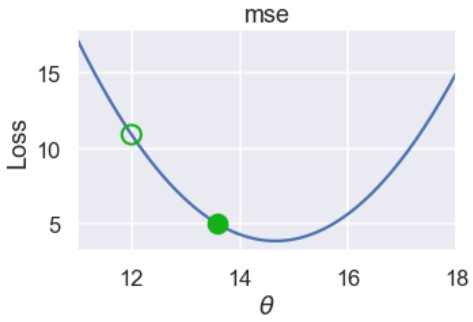
على الرغم أن θ انتقلت إلى الجانب الأيمن، لكنها لا زالت بعيده جداً عن القيمة الدنيا. يمكننا حل ذلك عن طريق ضرب الميلان بقيمه صغيرة قبل طرحه من θ. والعملية الحسابية النهاية ستبدو كالتالي:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \frac{\partial}{\partial \theta} L(\theta^{(t)}, \mathbf{y})$$

وفيها α هي قيمه ثابتة صغيرة. مثلاً، إذا حددنا قيمة $\alpha = 0.3$، فإن القيمة الجديدة ل $\theta^{(t+1)}$ ستكون:

```
plot_one_gd_iter(pts, 12, mse, grad_mse)
```

```
old theta: 12
new theta: 13.596
```



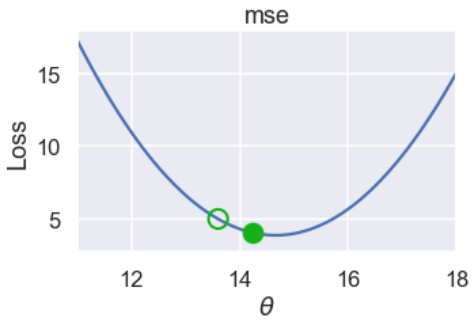
عرف الكاتب دالة جديده بأسم `plot_one_gd_iter` تقوم بكتابة ورسم بياني لقيمه θ السابقة والجديدة بعد القيام بالعملية الحسابية المشروحة مسبقاً، استخدم الكاتب أيضاً دالة بأسم `grad_mse` وهي تعريف لدالة `mse` باستخدام النزول الاشتقاقي، الكود البرمجي لكلا الدالتين:

```
def plot_one_gd_iter(y_vals, theta, loss_fn, grad_loss, alpha=0.3):
   new_theta = theta - alpha * grad_loss(theta, y_vals)
   plot_loss(pts, (11, 18), loss_fn)
   plot_theta_on_loss(pts, theta, loss_fn, c='none',
                      edgecolor=sns.xkcd_rgb['green'], linewidth=2)
   plot_theta_on_loss(pts, new_theta, loss_fn)
   print(f'old theta: {theta}')
   print(f'new theta: {new_theta}')

def grad_mse(theta, y_vals):
   return -2 * np.mean(y_vals - theta)
```

في الرسم التالي، قيم θ بعد عدة تكرارات بنفس الطريقة السابقة. لاحظ أن θ تتغير بشكل بسيط كلما اقتربنا من القيمة الدنيا لأن الميلان أيضاً أصبحت قيمته أقل:

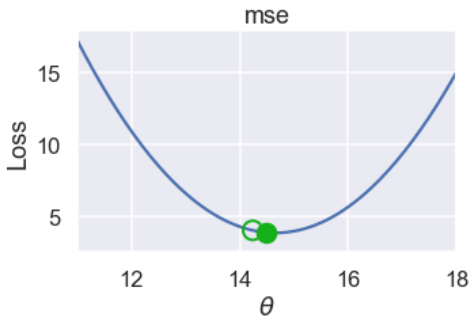
```
plot_one_gd_iter(pts, 13.60, mse, grad_mse)
```

```
old theta: 13.6
new theta: 14.236
```



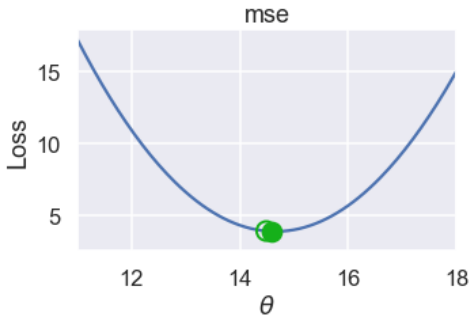

```
plot_one_gd_iter(pts, 14.24, mse, grad_mse)
```

```
old theta: 14.24
new theta: 14.492
```




```
plot_one_gd_iter(pts, 14.49, mse, grad_mse)
```

```
old theta: 14.49
new theta: 14.592
```



تحليل النزول الاشتقاقي

لدينا الآن فكرة عن طريقة عمل خوارزمية النزول الاشتقاقي:

- اختيار قيمه أوليه ل θ (في العادة تكون 0).
- إجراء العملية الحسابية $\theta - \alpha \cdot \frac{\partial}{\partial \theta} L(\theta, \mathbf{y})$ عليها وحفظ النتيجة كقيمه جديده ل θ.
- تكرار العملية حتى تتوقف θ عن التغير.

غالباً ستلاحظ استخدام رمز النزول (الانحدار) ∇_θ بدلاً من الاشتقاق الجزئي $\frac{\partial}{\partial \theta}$. كلا الرمزين متشابهان، ولكن بما أنا استخدام رمز النزول أكثر بشكل عام، فسنقوم باستخدامه في المعادلة:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \nabla_\theta L(\theta^{(t)}, \mathbf{y})$$

لمراجعة الرموز:

- $\theta^{(t)}$ هي التوقع الحالي ل θ^* في التكرار t.
- $\theta^{(t+1)}$ القيمة التالية ل θ.
- α يطلق عليها معدل التعلّم Learning Rate، وعادة ما تكون رقم صغير ثابت. في بعض المرات من المفيد أن تبدأ برقم عالي ل α والتقليل منه. إذا تغيرت قيمة α بين عمليات التكرار، نستخدم الرمز α^t لتوضيح تغير α في t.
- $\nabla_\theta L(\theta^{(t)}, \mathbf{y})$ هي اشتقاق جزئي لدالة الخسارة فيها قيمه متوقعة ل θ في التكرار t.

يمكننا ملاحظة أهمية استخدام دالة خسارة قابله للتفاضل: $\nabla_\theta L(\theta, \mathbf{y})$ هي جزء مهم من خوارزمية النزول الاشتقاقي. (على الرغم أن بالإمكان توقع قيمة النزول (الإنحدار) بحساب الفرق في الخسارة بين قيمتين θ وقسمتها على المسافه بينهما، لكن ذلك يزيد من مدة إيجاد النتيجة للنزول الاشتقاقي بشكل كبير مما يجعلها غير مفيده للاستخدام).

خوارزمية النزول الاشتقاقي بسيطه ومفيده بشكل كبير وذلك لإن بإمكاننا استخدامها في كثير من انواع النماذج والكثير من دوال الخسارة. هي الطريقة الحسابيه الأهم لضبط النماذج، بما فيها الإنحدار الخطي على بيانات بحجم كبير والشبكات العصبيه.

تعريف دالة `minimize`

الآن نعود لمهمتنا الأساسية: تعريف دالة `minimize`. سنحتاج للتعديل قليلاً من تعريف الدالة كوننا نريد إيجاد النزول الاشتقاقي لدالة الخسارة:

```
def minimize(loss_fn, grad_loss_fn, dataset, alpha=0.2, progress=True):
    '''
    تستخدم النزول الاشتقاقي للتقليل من دالة الخسارة loss_fn.
    عندما يكون theta_hat (θ^) تنتج لنا الداله القيمه الصغرى ل
    التغيير اقل من 0.001 بين التكرارات.
    '''
    theta = 0
    while True:
        if progress:
            print(f'theta: {theta:.2f} | loss: {loss_fn(theta, dataset):.2f}')
        gradient = grad_loss_fn(theta, dataset)
        new_theta = theta - alpha * gradient

        if abs(new_theta - theta) < 0.001:
            return new_theta

        theta = new_theta
```

ثم يمكننا تعريف دوال تقوم بحساب MSE و نزولها (انحدارها):

```
def mse(theta, y_vals):
    return np.mean((y_vals - theta) ** 2)

def grad_mse(theta, y_vals):
    return -2 * np.mean(y_vals - theta)
```

أخيراً، يمكننا استخدام الدالة `minimize` لحساب قيمة θ الأدنى للبيانات التالية $\mathbf{y} = [12.1, 12.8, 14.9, 16.3, 17.2]$

```
%%time
theta = minimize(mse, grad_mse, np.array([12.1, 12.8, 14.9, 16.3, 17.2]))
print(f'Minimizing theta: {theta}')
print()
```

```
theta: 0.00 | loss: 218.76
theta: 5.86 | loss: 81.21
theta: 9.38 | loss: 31.70
theta: 11.49 | loss: 13.87
theta: 12.76 | loss: 7.45
theta: 13.52 | loss: 5.14
theta: 13.98 | loss: 4.31
theta: 14.25 | loss: 4.01
theta: 14.41 | loss: 3.90
theta: 14.51 | loss: 3.86
theta: 14.57 | loss: 3.85
theta: 14.61 | loss: 3.85
theta: 14.63 | loss: 3.84
theta: 14.64 | loss: 3.84
theta: 14.65 | loss: 3.84
theta: 14.65 | loss: 3.84
theta: 14.66 | loss: 3.84
theta: 14.66 | loss: 3.84
Minimizing theta: 14.658511131035242

CPU times: user 7.88 ms, sys: 3.58 ms, total: 11.5 ms
Wall time: 8.54 ms
```

نلاحظ أن النزول الاشتقاقي قام بإيجاد نفس النتيجة بشكل سريع ل:

```
np.mean([12.1, 12.8, 14.9, 16.3, 17.2])
```

```
14.66
```

تقليل خسارة Huber

الآن، يمكننا تطبيق النزول الاشتقاقي للتقليل من دالة الخسارة Huber على بيانات الإكراميات.

```
tips = sns.load_dataset('tips')
tips['pcttip'] = tips['tip'] / tips['total_bill'] * 100
```

دالة الخسارة Huber تعرف كالتالي:

$$L_\delta(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} \frac{1}{2}(y_i - \theta)^2 & |y_i - \theta| \leq \delta \\ \delta(|y_i - \theta| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

والنزول الاشتقاقي لدالة Huber:

$$\nabla_\theta L_\delta(\theta, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} -(y_i - \theta) & |y_i - \theta| \leq \delta \\ -\delta \cdot \text{sign}(y_i - \theta) & \text{otherwise} \end{cases}$$

(لاحظ أننا في التعاريف السابقة لدالة خسارة Huber استخدمنا المتغير α للإشارة لنقطة الانتقال. ولإبعاد الشك بينها وبين α المستخدمة في النزول الاشتقاقي، قمنا بتغير رمز نقطة الانتقال في دالة الخسارة Huber إلى الرمز δ.)

```
def huber_loss(theta, dataset, delta = 1):
    d = np.abs(theta - dataset)
    return np.mean(
        np.where(d <= delta,
                 (theta - dataset)**2 / 2.0,
                 delta * (d - delta / 2.0))
    )

def grad_huber_loss(theta, dataset, delta = 1):
    d = np.abs(theta - dataset)
    return np.mean(
        np.where(d <= delta,
                 -(dataset - theta),
                 -delta * np.sign(dataset - theta))
    )
```

لنقوم بالتقليل من دالة الخسارة Huber في بيانات الإكرامية:

```
%%time
theta = minimize(huber_loss, grad_huber_loss, tips['pcttip'], progress=False)
print(f'Minimizing theta: {theta}')
print()
```

```
Minimizing theta: 15.506849531471964

CPU times: user 194 ms, sys: 4.13 ms, total: 198 ms
Wall time: 208 ms
```

ملخص النزول الاشتقاقي

يوفر لنا النزول الاشتقاقي طريقه عامله للتقليل من دالة الخسارة عندما لا نستطيع إيجاد القيمة الدنيا ل θ. عندما يكون نموذجنا ودالة الخسارة أكثر تعقيداً، نستخدم النزول الاشتقاقي كوسيلة لضبط النماذج.

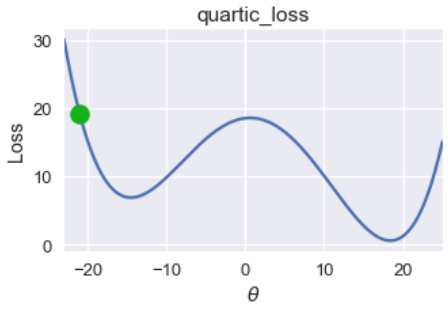
التحدب

يساهم النزول الاشتقاقي بشكل عام بالتقليل من دالة الخسارة. كما اظهرنا ذلك في دالة Huber للخسارة، تكمن فائدة النزول الاشتقاقي عندما يكون صعب علينا إيجاد القيمة الدنيا.

اكتشاف الحدود الدنيا بالنزول الاشتقاقي

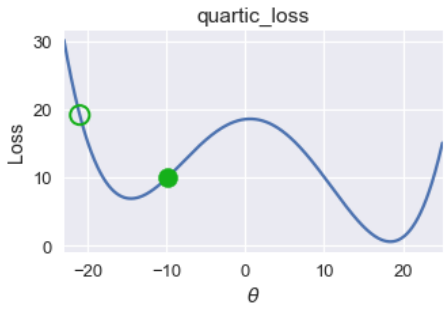
للأسف، في بعض الأحيان لا يمكن للنزول الاشتقاقي إيجاد قيمة θ. لنفترض التالي $\theta = -21$:

```
pts = np.array([0])
plot_loss(pts, (-23, 25), quartic_loss)
plot_theta_on_loss(pts, -21, quartic_loss)
```




```
plot_one_gd_iter(pts, -21, quartic_loss, grad_quartic_loss)
```

```
old theta: -21
new theta: -9.944999999999999
```

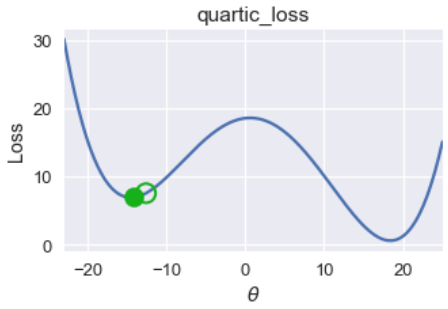



```
plot_one_gd_iter(pts, -9.9, quartic_loss, grad_quartic_loss)
```

```
old theta: -9.9
new theta: -12.641412
```

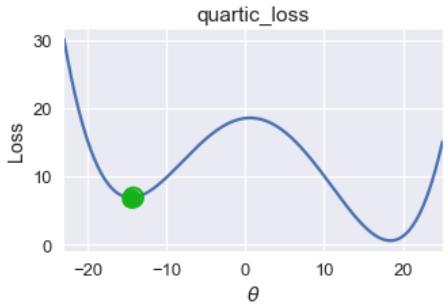

```
plot_one_gd_iter(pts, -12.6, quartic_loss, grad_quartic_loss)
```

```
old theta: -12.6
new theta: -14.162808
```




```
plot_one_gd_iter(pts, -14.2, quartic_loss, grad_quartic_loss)
```

```
old theta: -14.2
new theta: -14.497463999999999
```



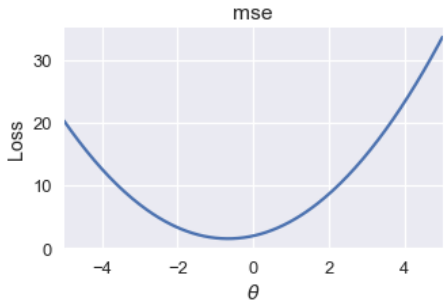
في الأمثلة السابقة استخدم الكاتب الدوال التالية (دالة الخسارة الرباعية ونزولها الاشتقاقي) التي لم يعرفها مسبقاً:

```
def quartic_loss(theta, y_vals):
    return np.mean(1/5000 * (y_vals - theta + 12) * (y_vals - theta + 23)
                   * (y_vals - theta - 14) * (y_vals - theta - 15) + 7)

def grad_quartic_loss(theta, y_vals):
    return -1/2500 * (2 *(y_vals - theta)**3 + 9*(y_vals - theta)**2
                      - 529*(y_vals - theta) - 327)
```

في المثال السابق، دالة الخسارة الرباعية، كانت نتيجة النزول الاشتقاقي قريبه إلى $\theta = -14.5$، ولكن القيمة الدنيا العامة لدالة الخسارة هي $\theta = 18$، في هذا المثال نرى أن النزول الاشتقاقي يبحث عن *القيمة الدنيا المحلية Local Minimum* والتي ليست دائماً تساوي قيمة الخسارة *للقيمة الدنيا العامة Global Minimum*.

لحسن الحظ، بعض دوال الخسارة لديها نفس الرقم للقيمة الدنيا المحلية والعامة. لنأخذ مثلاً دالة الخطأ التربيعي المتوسط MSE:

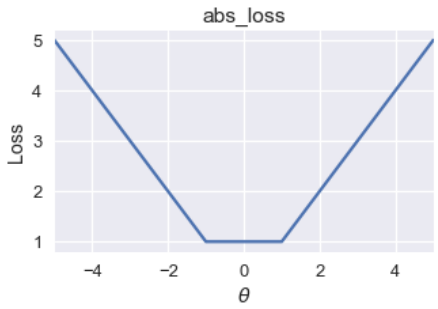
```
pts = np.array([-2, -1, 1])
plot_loss(pts, (-5, 5), mse)
```



تطبيق النزول الاشتقاقي على هذه الدالة سيوجد لنا دائماً قيمه مثاليه عامه ل θ كون القيمة الدنيا المحلية الوحيدة هي نفسها العامة.

متوسط الخطأ الحتمي قد يحتوي على أكثر من قيمه دنيا محلية. ولكن، كل القيم الدنيا هي نفسها القيمة الدنيا العامة.

```
pts = np.array([-1, 1])
plot_loss(pts, (-5, 5), abs_loss)
```



تعريف دالة متوسط الخطأ الحتمي

```
def abs_loss(theta, y_vals):
    return np.mean(np.abs(y_vals - theta))
```

في هذا المثال، ستكون قيمة النزول الاشتقاقي إحدى القيم المحلية الدنيا بين $[-1, 1]$ كون أن كل القيم في هذا النطاق قيم دنيا لدالة الخسارة هذه، سيقترح النزول الاشتقاقي قيمة دنيا مثاليه بين هذه النقاط ل θ.

تعريف التحدب

في بعض الدوال، أي قيمه دنيا محلية هي نفسهاالعامه. هذه الدوال يطلق عليها **دوال محدبة Convex function** لأنها منحنية للأعلى. كذلك دالة Huber للخسارة، النموذج الثابت، MSE و MAE جميعها محدبة.

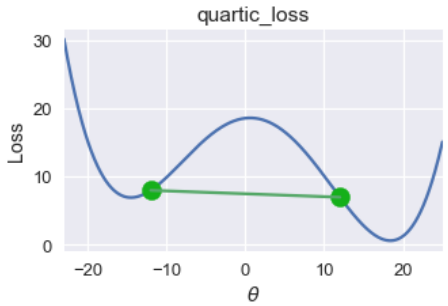
مع معدل تعلم Learning Rate مناسب، النزول الاشتقاقي يوجد θ العامة المثالية لدالة الخسارة المحدبة. وبسبب ذلك، نفضل ضبط نماذجنا باستخدام الدوال المحدبة إلا إذا كان لدينا سبب مناسب غير ذلك.

بشكلٍ عام، الدالة f يطلق عليها دالة محدبة فقط إذا كانت توفي شروط المتباينة لجميع مدخلاتها a و b ، لكل $t \in [0, 1]$: 📝

$$tf(a) + (1-t)f(b) \geq f(ta + (1-t)b)$$

المتباينة تقول إن جميع الخطوط التي تربط نقطتين في الدالة يجب أن تقع على أو فوق الدالة. لدالة الخسارة التي عرضناها في بداية هذا الجزء، يمكننا إيجاد هذه الخط:

```
pts = np.array([0])
plot_loss(pts, (-23, 25), quartic_loss)
plot_connected_thetas(pts, -12, 12, quartic_loss)
```



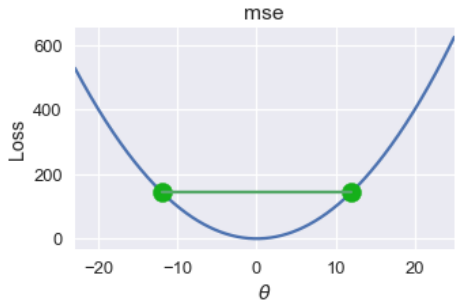
استخدم الكاتب الدالة plot_connected_thetas لتساعده على رسم الخط بين النقطتين، وعرفها الكاتب كالتالي:

```
def plot_connected_thetas(y_vals, theta_1, theta_2, loss_fn, **kwargs):
    plot_theta_on_loss(y_vals, theta_1, loss_fn)
    plot_theta_on_loss(y_vals, theta_2, loss_fn)
    loss_1 = loss_fn(theta_1, y_vals)
    loss_2 = loss_fn(theta_2, y_vals)
    plt.plot([theta_1, theta_2], [loss_1, loss_2])
```

وبناءًا على التعريف السابق، فهذه الدالة غير محدبة.

للخطأ التربيعي المتوسط، جميع الخطوط التي نقطتين تظهر فوق الرسم البياني. يمكننا رسم إحداها كالتالي:

```
pts = np.array([0])
plot_loss(pts, (-23, 25), mse)
plot_connected_thetas(pts, -12, 12, mse)
```



التعريف الرياضي للتحدب يعطينا وصف دقيق لتحديد ما اذا كانت الداله محدبه أو لا. في هذا الكتاب، سنتجاهل الجزء الرياضي لإثبات التحدب ونكتفي بالقول ما اذا كانت دالة محدبه أو لا.

ملخص التحدب

للداله المحدبه، أي قيمه دنيا محليه هي نفسها عامه. ذلك يسهل للنزول الاشتقاقي إيجاد افضل المتغيرات للنماذج لأي دالة خسارة. على الرغم من انه بإمكاننا استخدام النزول الاشتقاقي لدوال الخسارة غير المحدبه لإيجاد قيم دنيا، تلك القيم الدنيا المحليه غير مضمون أنها دائماً هي القيم العامه المثاليه.

النزول الاشتقاقي العشوائي

مقدمة

في هذا الجزء، سنتحدث عن تعديل على النزول الاشتقاقي يجعله أكثر فائدة للبيانات ذات الحجم الكبير. هذا التعديل يطلق عليه خوارزمية **النزول الاشتقاقي العشوائي Stochastic Gradient Descent**.

بعد أن تعلمنا طريقة عمل النزول الاشتقاقي وتحديثه لقيمة θ باستخدام الاشتقاق لدالة الخسارة. بالذات استخدمنا المعادلة التالية:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \nabla_\theta L(\theta^{(t)}, \mathbf{y})$$

في هذه المعادلة:

- $\theta^{(t)}$ هي التوقع الحالي ل *θ في التكرار t.
- α هي معدل التعلّم Learning Rate.
- $\nabla_\theta L(\theta^{(t)}, \mathbf{y})$ هي اشتقاق دالة الخسارة.
- ونحسب التوقع التالي $\theta^{(t+1)}$ عن طريق طرح α و $\nabla_\theta L(\theta, \mathbf{y})$ والمحسوبة في $\theta^{(t)}$.

حدود النزول الاشتقاقي

في المعادلة السابقة، قمنا بحساب $\nabla_\theta L(\theta, \mathbf{y})$ باستخدام متوسط الاشتقاق لدالة الخسارة $\ell(\theta, y_i)$ لجميع البيانات. بمعنى آخر، في كل مرة نحدث قيمة θ نتحقق من جميع النقاط الأخرى في بياناتنا. لهذا السبب، القانون للاشتقاق في المعادلة السابقة يطلق عليه **النزول الاشتقاقي المُجَمع Batch Gradient Descent**.

ولأننا لسوء الحظ عادة ما نعمل مع بيانات كبيرة الحجم، فإن النزول الاشتقاقي المُجَمع سيعمل لإيجاد القيمة المناسبة ل θ بعد بضع تكرارات، ولكن كل تكرار قد يأخذ وقتاً طويل لحساب النتيجة فيه إذا كانت النقاط في بياناتنا كثيره.

النزول الاشتقاقي العشوائي

لحل مشكلة الوقت في حساب الاشتقاق لجميع بيانات التدريب، يقوم النزول الاشتقاقي العشوائي بتوقع القيمة باستخدام **قيمه عشوائية واحدة من البيانات**. ولأن القيمة تم اختيارها بشكل عشوائي، نتوقع أن الاشتقاق لكل نقطة سيؤدي بالنهاية إلى نفس النتيجة للنزول الاشتقاقي المُجَمع.

لنعود مرة أخرى لمعادلة النزول الاشتقاقي المُجَمع:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \nabla_\theta L(\theta^{(t)}, \mathbf{y})$$

في هذه المعادلة، لدينا المصطلح $\nabla_\theta L(\theta^{(t)}, \mathbf{y})$، متوسط الاشتقاق لدالة الخسارة بين كل النقاط في بيانات التدريب، ومعادلتها:

$$\nabla_\theta L(\theta^{(t)}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(\theta^{(t)}, y_i)$$

وفيها $\ell(\theta, y_i)$ هي الخسارة في نقطة معينة في بيانات التدريب. لتطبيق النزول الاشتقاقي العشوائي، ببساطه نقوم بتغير متوسط الاشتقاق ب الاشتقاق في نقطة معينة. المعادلة بعد التعديل للنزول الاشتقاقي العشوائي هي:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \nabla_\theta \ell(\theta^{(t)}, y_i)$$

في هذه المعادلة، y_i يتم اختيارها بشكل عشوائي من $\mathbf{y}$. لاحظ أن اختيار النقاط بشكل عشوائي مهم جداً لنجاح النزول الاشتقاقي العشوائي! إذا لم يتم اختيار النقاط بشكل عشوائي، قد ينتج لنا نتائج أسوأ من نتائج النزول الاشتقاقي المُجَمع.

نقوم عادةً باستخدام النزول الاشتقاقي العشوائي عن طريق خلط البيانات واستخدام كل نقطة بعد الخلط حتى تتجاوز أحد النقاط بيانات التدريب. إذا لم يتم ذلك، نعيد خلط النقاط والقيام بنفس الخطوات حتى تتجاوز البيانات. في كل **تكرار Iteration** النزول الاشتقاقي العشوائي يتحقق من نقطة واحده؛ وكل عملية تجاوز تتم بنجاح يطلق عليها **Epoch**.

استخدام دالة الخسارة MSE

كمثال، لنطبق النزول الاشتقاقي العشوائي على دالة الخسارة لMSE. لنتذكر تعريفها:

$$L(\theta, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta)^2$$

أخذ الاشتقاق θ يصبح لدينا:

$$\nabla_\theta L(\theta, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} -2(y_i - \theta)$$

بما أن المعادلة السابقة تعطينا متوسط خسارة الاشتقاق لكل النقاط في البيانات، فإن خسارة الاشتقاق لنقطة معينة هي ببساطه جزء المعادلة التي تم أخذ متوسطه:

$$\nabla_\theta \ell(\theta, y_i) = -2(y_i - \theta)$$

لتحديثها للاشتقاق المُجَمع لدالة الخسارة MSE:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot \left(\frac{1}{n} \sum_{i=1}^{n} -2(y_i - \theta) \right)$$

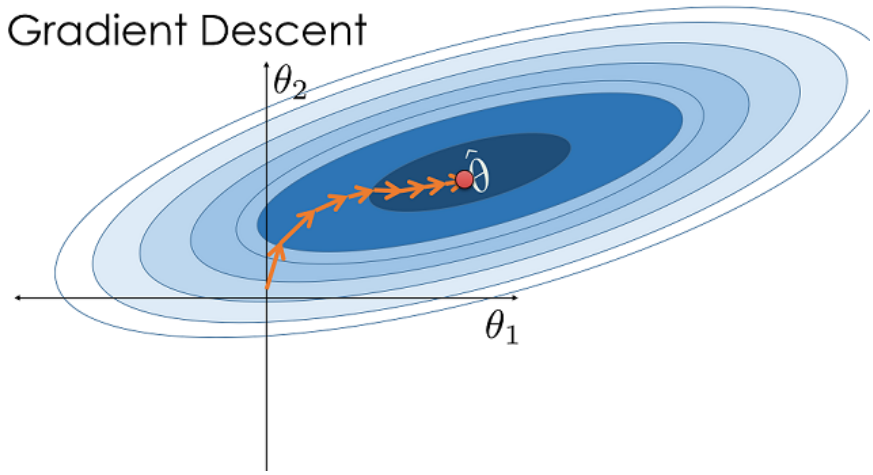
والنزول الاشتقاقي العشوائي لها سيكون كالتالي:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \cdot (-2(y_i - \theta))$$

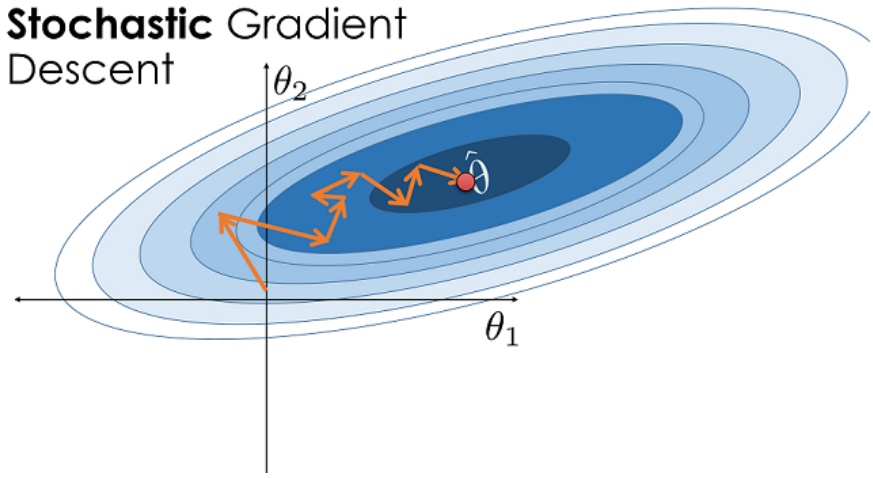
سلوك النزول الاشتقاقي العشوائي

بما أن الاشتقاق العشوائي فقط يتحقق من نقطة واحدة كل مرة، فإن تحديثه لقيمة θ سيكون أقل دقه من التحديث إذا تم من النزول العشوائي المُجَمع. ولكن، بما أنه أسرع في حساب النتائج، فإن النزول الاشتقاقي العشوائي بإمكانه التقدم بشكل كبير للوصول للقيمة المناسبة ل θ في حين النزول

العشوائي المُجَمع لم ينتهي وقتها من التحديث ولا مرة واحدة. الصورة في الأسفل توضح تحديثات تمت بنجاح لقيمة θ باستخدام النزول الاشتقاقي المُجَمع. المساحة غامقة اللون في الصوره تعني القيمة المثالية ل θ على بيانات التدريب، وهي $\hat{\theta}$. (الصورة تظهر نموذج لديه متغيران، ولكن من المهم ملاحظة طريقة النزول الاشتقاقي المُجَمع التي يصل فيها ل $\hat{\theta}$.)



في الجانب الآخر، النزول الاشتقاقي العشوائي، عادة ما يأخذ خطوه بعيداً عن $\hat{\theta}$، ولكن كونه يقوم بالتحديث بشكل متكرر، يصل غالباً للنقطة المثالية بشكل أسرع من المُجَمع.



تعريف دالة للنزول الاشتقاقي العشوائي

كما فعلنا سابقاً، نقوم بتعريف دالة تحسب لنا النزول الاشتقاقي العشوائي لدالة خسارة. ستكون مشابهة لدالة `minimize` التي سبق تعريفها، ولكن نحتاج لإضافة الاختيار العشوائي للقيم في كل تكرار:

```
def minimize_sgd(loss_fn, grad_loss_fn, dataset, alpha=0.2):
    """
    تستخدم النزول الاشتقاقي العشوائي للتقليل من دالة الخسارة loss_fn
    عندما يكون θ تكون النتيجة القيمه الصغرى ل
    الفرق بين التكرارات اقل من 0.001
    """
    NUM_OBS = len(dataset)
    theta = 0
    np.random.shuffle(dataset)
    while True:
        for i in range(0, NUM_OBS, 1):
            rand_obs = dataset[i]
            gradient = grad_loss_fn(theta, rand_obs)
            new_theta = theta - alpha * gradient

            if abs(new_theta - theta) < 0.001:
                return new_theta

            theta = new_theta
        np.random.shuffle(dataset)
```

نزول اشتقاقي بدفعات صغيرة

النزول الاشتقاقي بدفعات صغيرة Mini-batch Gradient Descent يحاول أن يوازن بين النزول الاشتقاقي العشوائي و المُجَمع عن طريقه زيادة عدد الأرقام التي يتطلع عليها في كل عملية تكرار. في النزول الاشتقاقي بدفعات صغيرة، نستخدم عدد من النقاط في كل تحديث بدلاً من نقطة واحدة. نستخدم متوسط الاشتقاق لدوال الخسارة للقيام بتوقع لقيمة الاشتقاق الصحيحة لخسارة الانتروبيا التقاطعية Cross Entropy Loss. إذا كانت $\mathcal{B}$ هي الدفعة الصغيرة من النقاط التي نختارها بشكل عشوائي من n، فالمعادلة الحالية كالتالي:

$$\nabla_\theta L(\theta, \mathbf{y}) \approx \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_\theta \ell(\theta, y_i)$$

كما في النزول الاشتقاقي العشوائي، نقوم بالنزول الاشتقاقي بدفعات صغيرة عن طريق خلط بيانات التدريب واختيار دفعات عن طريق التكرار داخل البيانات المخلوطة. بعد كل Epoch، نعيد خلط البيانات واختيار دفعه صغيرة جديده.

على الرغم من أننا فرقنا بين النزول الاشتقاقي العشوائي وبدفعات صغيرة في هذه الكتاب، يستخدم مصطلح النزول الاشتقاقي العشوائي بشكل عام للاشتقاقات بدفعات صغيرة بأي حجم.

اختيار حجم الدفعات الصغيرة

يكون النزول الاشتقاقي بدفعات صغيرة مثالياً عندما يعمل على وحدة المعالجة الرسوميه GPU (كرت الشاشة للحاسب). كون العمليات من هذا النوع تأخذ وقتاً طويل، استخدام الدفعات الصغيرة يزيد من دقة الاشتقاق دون الزيادة من سرعة عملية الحساب. بناءًا على ذاكرة وحدة المعالجة الرسومية في الجهاز، يتم تحديد حجم الدفعات بين 10 و 100.

تعريف دالة للنزول الاشتقاقي بدفعات صغيرة

دالة النزول الاشتقاقي بدفعات صغيرة تحتاج لخيار لتحديد حجم الدفعات. الدالة التالية توفر هذه الخاصية:

```
def minimize_mini_batch(loss_fn, grad_loss_fn, dataset, minibatch_size, alpha=0.2):
    """
    loss_fn تستخدم النزول الاشتقاقي العشوائي بدفعات صغيره للتقليل من دالة الخسارة
    عندما يكون θ تكون النتيجة القيمه الصغرى ل
    0.001 الفرق بين التكرارات اقل من
    """
    NUM_OBS = len(dataset)
    assert minibatch_size < NUM_OBS

    theta = 0
    np.random.shuffle(dataset)
    while True:
        for i in range(0, NUM_OBS, minibatch_size):
            mini_batch = dataset[i:i+minibatch_size]
            gradient = grad_loss_fn(theta, mini_batch)
            new_theta = theta - alpha * gradient

            if abs(new_theta - theta) < 0.001:
                return new_theta

            theta = new_theta
        np.random.shuffle(dataset)
```

ملخص النزول الاشتقاقي العشوائي

نستخدم النزول الاشتقاقي بدفعات لتحسين النموذج بشكل متكرر حتى نصل إلى القيمة الدنيا للخسارة. بما أن النزول الاشتقاقي بدفعات يكون صعباً للحساب في البيانات الكبيرة، عادةً ما نستخدم النزول الاشتقاقي العشوائي لضبط تلك النماذج. عند استخدام GPU، النزول الاشتقاقي بدفعات بسيطه يمكنه الحساب بشكل أسرع من العشوائي بنفس تكلفة التشغيل. للبيانات الكبيرة، النزول الاشتقاقي العشوائي وبدفعات صغيرة هي أفضل للاستخدام كونها أسرع للحساب.

الاحتماليات والتعميم

مقدمة

تعرفنا على خطوات إنشاء نماذج باستخدام البيانات:

- اختيار النموذج.
- اختيار دالة الخسارة.
- ضبط النموذج عن طريق تقليل الخسارة.

حتى الآن، تعرفنا على النموذج الثابت، عدد مختلف من دوال الخسارة، والنزول الاشتقاقي كوسيلة عامه للتقليل من الخسارة. إتباع هذه الخطوات عادةً ما يكوّن لنا نموذج يقوم بتوقعات صحيحه على البيانات التي تدرب عليها.

للأسف، فائدة النموذج الذي يأتي بشكل جيد على بيانات التدريب عند تطبيقه على بيانات مختلفة محدود وقليل. نهتم بإمكانية النموذج على **التعميم**. نحتاج أن يكون نموذجنا قادر على التوقع بشكل صحيح بشكل عام، ليس فقط على بيانات التدريب. هذه المشكلة قد تبدو صعبة، قد تتساءل كيف يمكننا الإجابة وتوقع نتائج بيانات لم نرها بعد؟

هنا نستعين بالاستنتاجات في الإحصاء. سنتعرف أولاً على بعض الأدوات الرياضية: العينات العشوائية، التوقع والتباين. باستخدام هذه الأدوات، سنتمكن من إيجاد توقعات مستقبليه لأداء النموذج على بياناتنا، وحتى تلك التي لم يتدرب النموذج عليها!

المتغيرات العشوائية

تحتوي جميع الظواهر في العالم الحقيقي على القليل من العشوائية، مما يجعل تكوين وجمع البيانات عشوائياً بشكل طبيعي. لأننا نضبط نماذجنا على هذه البيانات، فإن النماذج أيضاً تحتوي على بعض العشوائية. للتعبير عن هذه العشوائية بشكل رياضي، نستخدم المتغيرات العشوائية.

المتغير العشوائي Random Variable هو متغير جبري يمثل **قيمه رقمية / عدديه تم تحديديها بواسطة الاحتمالات**. في هذا الكتاب، سنستخدم دائماً الحروف الكبيرة مثل X و Y لتمثيل متغير عشوائي. على الرغم أن المتغيرات العشوائية يمكن تمثيلها بعدة أشكال كقيم منفصله (مثلاً عدد الذكور في عينة من 10 أشخاص) أو كقيم متسلسلة (مثلاً متوسط درجة الحرارة في لوس أنجلوس)، في هذا الكتاب، سنستخدم المتغيرات العشوائية المنفصلة. 📝📝

يجب علينا دائماً تحديد ما يمثله المتغير العشوائي. مثلاً، قد نكتب أن المتغير العشوائي X يمثل عدد مرات ظهور صورة في 10 مرات من رمي العملة المعدنية. تعريف المتغير العشوائي يحدد القيم التي بداخله. في المثال السابق، X ستأخذ فقط القيم من 0 إلى 10.

يجب علينا أيضاً تحديد احتمالية ظهور أي من القيم المحتملة في المتغير العشوائي. مثلاً، احتمالية أن $X = 0$ يتم كتابتها كالتالي $P(X = 0) = (0.5)^{10}$ كذلك وبنفس الطريقة يمكننا حساب احتمالية أن X هي أي قيمه من بين القيم التالية $\{0, 1, \ldots, 10\}$.

دالة كتلة الاحتمال

دالة كتلة الاحتمال Probability Mass Function (PMF) أو **توزيع Distribution** المتغير العشوائي X تقدم لنا احتمالية أن X هي أحد القيم المحتملة. إذا جعلنا $\mathbb{X}$ ترمز لمصفوفة القيم التي يمكن أن تمثلها X و x قيمه معينة من القيم في $\mathbb{X}$، فإن دالة كتلة الاحتمال ل X يجب أن توفي الشروط التالية: 📝

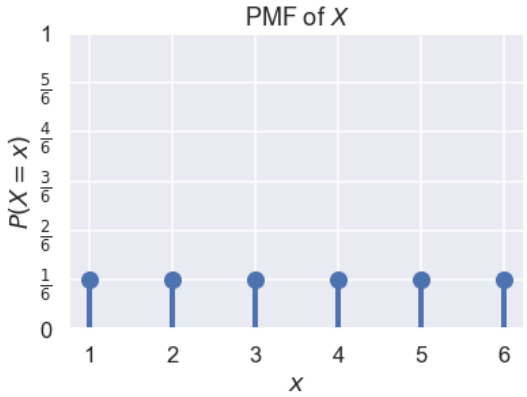
$$1) \sum_{x \in \mathbb{X}} P(X = x) = 1$$

$$2) \text{ For all } x \in \mathbb{X}, 0 \leq P(X = x) \leq 1$$

الشرط الأول يقول إن مجموع الاحتماليات للقيم فيه X يساوي 1.

الشرط الثاني يقول إن احتمالية كل قيمه من القيم في X يجب أن تكون بين 0 و 1.

لنفترض أن X تمثل نتيجة رمي حجر النرد. نعلم أن $X \in \{1, 2, 3, 4, 5, 6\}$ و أن $P(X = 1) = P(X = 2) = \ldots = P(X = 6) = \frac{1}{6}$. يمكننا رسم نتيجة PMF ل X كالتالي:



التوزيعات المشتركة

بشكل تلقائي، تمتد فكرة التوزيع PMF للمتغير العشوائي إلى التوزيع المشترك للعديد من المتغيرات العشوائية. بشكل أوضح، **التوزيع المشترك Joint Distribution** لمتغيران عشوائيان أو أكثر ينتج عنه احتماليه أن هذه المتغيرات العشوائية تأخذ نفس مصفوفة القيم. 📝

مثلاً، المتغير العشوائي X يمثل عدد مرات ظهور صوره عند رمي العملة المعدنية 10 مرات، و Y تمثل عدد مرات ظهور كتابه عند رمي العملة المعدنية 10 مرات. يمكننا ملاحظة أن:

$$P(X = 0, Y = 10) = P(X = 10, Y = 0) = (0.5)^{10}$$

بينما $P(X = 6, Y = 6) = 0$ لأنه منا لمستحيل أن نحصل على 6 صور و 6 كتابات في 10 رميات.

التوزيع الهامشي

أحياناً، نبدأ بتوزيع مشترك لمتغيران عشوائيان X و Y ولكن نريد البحث عن توزيع X فقط. يطلق على ذلك **التوزيع الهامشي Marginal Distribution**. لإيجاد احتمالية X لقيمه معينة، يجب علينا الأخذ بالاعتبار جميع القيم المحتملة ل Y والتي قد تأتي مع X وجمع كل هذه الاحتماليات المشتركة: 📝

$$\sum_{y \in \mathbb{Y}} P(X = x, Y = y) = P(X = x)$$

يمكننا إثبات ذلك على النحو التالي:

$$\begin{aligned}
\sum_{y \in \mathbb{Y}} P(X = x, Y = y) &= \sum_{y \in \mathbb{Y}} P(X = x) \times P(Y = y \mid X = x) \\
&= P(X = x) \times \sum_{y \in \mathbb{Y}} P(Y = y \mid X = x) \\
&= P(X = x) \times 1 \\
&= P(X = x)
\end{aligned}$$

في السطر الأخير من الإثبات، تعاملنا مع $Y \mid X = x$ كقيم عشوائية مع PMF غير معروفة. ذلك يهمنا لأننا استخدمنا الخاصية الأولى التي تقول إن مجموع الاحتماليات ل PMF يجب أن يساوي 1، يعني ذلك أن $\sum_{y \in \mathbb{Y}} P(Y = y \mid X = x) = 1$.

المتغيرات العشوائية المستقلة

كما في الأحداث، المتغيران العشوائيين قد يكونا مستقلان أو غير مستقلان. أي متغيرين عشوائيين يكونان مستقلان فقط اذا كانت معرفة نتيجة إحداهما لا تأثر على احتمالية أي نتيجة للمتغير الآخر.

مثلاً، عند رمي العملة المعدنية عشر مرات لنجعل X تكون عدد مرات ظهور الصورة و Y عدد مرات ظهور الكتابة. بشكل واضح، X و Y غير مستقلان كون معرفتنا أن $X = 0$ تعني أن $Y = 10$.

يمكننا بدلاً من ذلك إجراء جولتين لرمي العملة المعدنية عشر مرات. إذا كانت X هي عدد مرات ظهور صوره في الجولة الأولى و Y هي عدد مرات ظهور صوره في الجولة الثانية، فإن X و Y مستقلان لأن النتائج للجولة الأولى من رمي العملة المعدنية لا تأثر على النتائج من رمي العملة المعدنية في الجولة الثانية.

مثال على الأعمار

لنفترض أن لدينا البيانات البسيطة التالية لأربع أشخاص:

	Name	Age
0	Carol	50
1	Bob	52
2	John	51
3	Dave	50

لنفترض اننا أخذنا شخصين كعينه من هذه البيانات ووضعنا اشخاص بدلاً منهم. اذا كان المتغير العشوائي Z يمثل الفرق في الأعمار بين الشخص الأول والثاني في العينة، فما هو توزيع المتغير العشوائي **PMF** ل Z?

لحل هذه المشكله، نقوم من جديد بتعريف متغيران عشوائيان. نقوم بتعريف X كعمر الشخص الأول و Y كعمر للشخص الثاني. فأن $Z = X - Y$. ثم نوجد توزيع الإحتمالات المشتركة ل X و Y، والتي تعني الإحتماليه لكل متغير أن يحصل عليها بنفس الوقت. في هذه الحاله، نلاحظ أن X و Y مستقلان وموزعين بالتساوي؛ كلا المتغيران العشوائيان يمثلان احتملات عشوائية مستقله للإختيار من بين البيانات، وعملية الإختيار الأولى لا تأثر على الثانيه. مثلاً، إحتمالية أن $X = 51$ و $Y = 50$ هي $P(X = 51, Y = 50) = \frac{1}{4} \cdot \frac{2}{4} = \frac{2}{16}$. بنفس الطريقه، نحصل على التالي:

	$Y = 50$	$Y = 51$	$Y = 52$
$X = 50$	$\frac{4}{16}$	$\frac{2}{16}$	$\frac{2}{16}$
$X = 51$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
$X = 52$	$\frac{2}{16}$	$\frac{1}{16}$	$\frac{1}{16}$

دعنا الآن نأخذ بعين الاعتبار لو أخذنا شخصان من هذه البيانات دون توفير بدائل لهم. كما في السابق، نعرف X كالعمر للشخص الأول و Y العمر للشخص الثاني، و $Z = X - Y$. ولكن، الآن X و Y ليست مستقلة؛ مثلاً، إذا عرفنا أن $X = 51$، فإن $Y \neq 51$. ونجد التوزيع المشترك ل X و Y كالتالي:

	$Y = 50$	$Y = 51$	$Y = 52$
$X = 50$	$\frac{2}{12}$	$\frac{2}{12}$	$\frac{2}{12}$
$X = 51$	$\frac{2}{12}$	0	$\frac{1}{12}$
$X = 52$	$\frac{2}{12}$	$\frac{1}{12}$	0

يمكننا إيجاد التوزيع الهامشي ل Y من الجدول:

$$
\begin{aligned}
P(Y=50) &= P(Y=50, X=50) + P(Y=50, X=51) + P(Y=50, X=52) \\
&= \frac{2}{12} + \frac{2}{12} + \frac{2}{12} \\
&= \frac{1}{2} \\
P(Y=51) &= \frac{2}{12} + 0 + \frac{1}{12} = \frac{1}{4} \\
P(Y=52) &= \frac{2}{12} + \frac{1}{12} + 0 = \frac{1}{4}
\end{aligned}
$$

لاحظ أننا قمنا بجمع كل عمود من التوزيع المشترك في الجدول العلوي. يمكن للشخص حساب نتائج الجمع لكل عمود وكتابتها على الهامش أسفل الجدول؛ هذه كانت بداية فكرة المصطلح التوزيع الهامشي.

يجب أن تلاحظ أيضاً أن X و Y ليست مستقلة عندما نأخذ عينات دون توفير بدائل. مثلاً، إذا كانت $X = 52$ فإن $Y \neq 52$. مع ذلك، لا يزال التوزيع الهامشي بين X و Y نفسه.

ملخص المتغيرات العشوائية

في هذا الجزء، تعرفنا على المتغيرات العشوائية، متغيرات رياضية تأخذ قيم بناءًا على عملية عشوائية. هذه النتائج يجب أن تعرف بشكل كامل ودقيق، كل نتيجة يجب أن تحتوي على احتماليه معرفة بشكل واضح للحالات. المتغير العشوائي مفيد جداً في كثير من الحالات، بما فيها عملية جمع البيانات.

التوقع والتباين

على الرغم أن المتغير العشوائي يتم وصفحه بشكل كامل بواسطة دالة PMF الخاصة به، عادةً ما نستخدم **التوقع** و **التباين** لوصف متوسط وتوزيع المتغير على المدى الطويل. هذان المتغيران لديهما خصائص رياضية مميزة والتي لديها أهمية خاصة لعلم البيانات. مثلاً، يمكننا إظهار أن التوقع صحيح على المدى البعيد عن طريق إظهار نتيجة التوقع مساويه لقيمه مغلّمة المجتمع الإحصائي. سنبدأ بتعريف التوقع والتباين، والتعرف على أهم خصائصهم الرياضية، والانتهاء بتطبيق بسيط على التوقع. 📝

التوقع

بالعادة، يهمنا متوسط القيمة العشوائية على المدى البعيد لأنها تعطينا نظره عن توزيع البيانات. نطلق على المتوسط على المدى البعيد **القيمة المتوقعة Expected Value**، أو **التوقع Expectation** للقيمة العشوائية. التوقع القيمة العشوائية X هو:

$$
\mathbb{E}[X] = \sum_{x \in \mathbb{X}} x \cdot P(X = x)
$$

مثلاً، إذا كانت X تمثل رمي حجر النرد مرة واحدة:

$$
\begin{aligned}
\mathbb{E}[X] &= 1 \cdot P(X=1) + 2 \cdot P(X=2) + \ldots + 6 \cdot P(X=6) \\
&= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \ldots + 6 \cdot \frac{1}{6} \\
&= 3.5
\end{aligned}
$$

لاحظ أن القيمة المتوقعة ل X لا يجب أن تكون القيمة المحتملة لها. على الرغم أن $\mathbb{E}[X] = 3.5$، ولكن X لا يمكنها أن تكون القيمة 3.5

> القيمة 3.5 ليست موجودة في حجر النرد، القيمة المتوقعة هي من 1 حتى 6.

مثال: لنعود للبيانات في الجزء السابق.

	Name	Age
0	Carol	50
1	Bob	52
2	John	51
3	Dave	50

نختار شخصاً واحداً من هذه البيانات بشكل محايد وعشوائي. لتكون Y هي القيمة العشوائية التي تمثل عمر هذا الشخص. إذاً:

$$
\begin{aligned}
\mathbb{E}[Y] &= 50 \cdot P(Y=50) + 51 \cdot P(Y=51) + 52 \cdot P(Y=52) \\
&= 50 \cdot \frac{2}{4} + 51 \cdot \frac{1}{4} + 52 \cdot \frac{1}{4} \\
&= 50.75
\end{aligned}
$$

مثال: لنفرض أننا سنأخذ عينتين من البيانات مع استبدالهم. إذا كانت القيمة العشوائية Z تمثل الفرق في العمر بين الشخص الأول والثاني في هذه العينة، فما هي $\mathbb{E}[Z]$؟

كما في الجزء السابق، عرفنا X على أنا عمر الشخص الأول، و Y هي عمر الشخص الثاني، ف $Z = X - Y$. من التوزيع المشترك ل X و Y الذي أعطي في الجزء السابق، يمكننا إيجاد دالة كتلة الاحتمال ل Z، مثلاً:

$$P(Z = 1) = P(X = 51, Y = 50) + P(X = 52, Y = 51) = \frac{3}{16}$$

فإن:

$$\begin{aligned}\mathbb{E}[Z] &= (-2) \cdot P(Z = -2) + (-1) \cdot P(Z = -1) + \ldots + (2) \cdot P(Z = 2) \\ &= (-2) \cdot \frac{2}{16} + (-1) \cdot \frac{3}{16} + \ldots + (2) \cdot \frac{2}{16} \\ &= 0\end{aligned}$$

بما أن $\mathbb{E}[Z] = 0$، فأننا نتوقع على المدى البعيد أن الفرق في الأعمار في عينه ذات الحجم 2 يساوي 0.

خطية التوقع

عند التعامل مع مجموعات خطية للمتغيرات العشوائية كما رأينا في الأمثلة السابقة، يمكننا عادةً استخدام **خطية التوقع Linearity of Expectations** بدلاً من الحساب يدوياً للتوزيع المشترك. 📝

خطية التوقع يمكن وصفها كالتالي:

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

ومن ذلك الوصف يمكننا قول التالي:

$$\mathbb{E}[cX] = c\mathbb{E}[X]$$

وفيها X و Y هي متغيرات عشوائية، و c رقم ثابت.

يمكننا القول، التوقع لمجموع متغيرين عشوائيين هو مجموع التوقع للمتغيرات.

في المثال السابق، رأينا أن $Z = X - Y$. فإن:

$$\mathbb{E}[Z] = \mathbb{E}[X - Y] = \mathbb{E}[X] - \mathbb{E}[Y]$$

الآن يمكننا حساب $\mathbb{E}[X]$ و $\mathbb{E}[Y]$ بشكل منفصل عن بعضهما. لأن $\mathbb{E}[X] = \mathbb{E}[Y] = 50.75$ فإن $\mathbb{E}[Z] = 50.75 - 50.75 = 0$.

يمكن حساب خطية التوقع حتى لو كانت X و Y غير مستقلان! كمثال، لنأخذ شخصين من نفس البيانات في الجزء السابق دون توفير بديل لهم. كما في السابق، عرفنا X أنها عمر الشخص الأول و Y هي عمر الشخص الثاني، و $Z = X - Y$. بشكل واضح، X و Y غير مستقلان، معرفة أن $X = 52$، تعني أن $Y \neq 52$.

من خلال التوزيع المشترك ل X و Y الذي سبق أن أعطي في الجزء السابق، يمكننا إيجاد $\mathbb{E}[Z]$:

$$\begin{aligned}\mathbb{E}[Z] &= (-2) \cdot P(Z = -2) + (-1) \cdot P(Z = -1) + \ldots + (2) \cdot P(Z = 2) \\ &= (-2) \cdot \frac{2}{12} + (-1) \cdot \frac{3}{12} + \ldots + (2) \cdot \frac{2}{12} \\ &= 0\end{aligned}$$

طريقة أسهل لإيجاد التوقع هي استخدام خطية التوقع. حتى ولو كانت X و Y مستقلان، فإن $\mathbb{E}[Z] = \mathbb{E}[X - Y] = \mathbb{E}[X] - \mathbb{E}[Y]$. لنتذكر من الجزء السابق أن X و Y لديهما نفس قيمة PMF حتى عندما قمنا بأخذ عينات دون توفير بدائل، والذي يعني أن $\mathbb{E}[X] = \mathbb{E}[Y] = 50.75$. وبالتالي، فكما في المثال الأول، $\mathbb{E}[Z] = 0$.

لاحظ أن خطية التوقع فقط تنطبق على القيم الخطية من المتغيرات العشوائية. مثلاً، $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ ليست خطيه ل X و Y. في هذه الحالة، $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ هي صحيحه بشكل عام فقط للمتغيرات العشوائية المستقلة.

التباين

التباين لمتغير عشوائي هي تعبير رقمي لتوزيع المتغير. مثلاً، للمتغير العشوائي X:

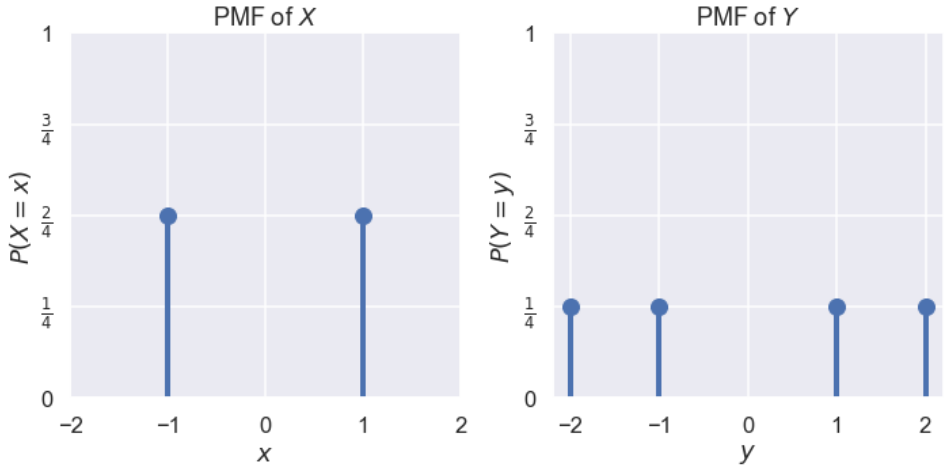
$$Var(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$$

المعادلة السابقة تقول إن التباين ل X هو تربيع متوسط المسافة للقيمة المتوقعة ل X.

مع بعض العمليات الرياضية التي لم نكتبها للاختصار، يمكننا أن نكتب المعادلة بالشكل التالي:

$$Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

لنأخذ المتغيرين العشوائيين X و Y ولديهما التوزيعات الاحتمالية التالية:



تأخذ X القيم $1-$ و 1 مع احتمالية $\frac{1}{2}$ لهما. بينما تأخذ Y القيم $2-$، $1-$، 1 و 2 مع احتمالية $\frac{1}{4}$ لكل قيمه. وجدنا أن $\mathbb{E}[X] = \mathbb{E}[Y] = 0$. لأن توزيع Y أعلى من X، فأننا نتوقع أن $Var(Y)$ أكبر من $Var(X)$.

$$\begin{aligned} Var(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= \mathbb{E}[X^2] - 0^2 \\ &= \mathbb{E}[X^2] \\ &= (-1)^2 P(X = -1) + (1)^2 P(X = 1) \\ &= 1 \cdot 0.5 + 1 \cdot 0.5 \\ &= 1 \end{aligned}$$

$$\begin{aligned} Var(Y) &= \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \\ &= \mathbb{E}[Y^2] - 0^2 \\ &= \mathbb{E}[Y^2] \\ &= (-2)^2 P(Y = -2) + (-1)^2 P(Y = -1) + (1)^2 P(Y = 1) + (2)^2 P(Y = 2) \\ &= 4 \cdot 0.25 + 1 \cdot 0.25 + 1 \cdot 0.25 + 4 \cdot 0.25 \\ &= 2.5 \end{aligned}$$

كما توقعنا، التباين في Y أكبر من التباين في X.

لدى التباين خاصية مفيده لتسهل من بعض العمليات الرياضية. إذا كانت X متغير عشوائي:

$$Var(aX + b) = a^2 Var(X)$$

إذا كان لدينا متغيرين عشوائيين X و Y مستقلان:

$$Var(X + Y) = Var(X) + Var(Y)$$

لاحظ أن خطية التوقع يمكن تطبيقها على X و Y حتى لو كانا غير مستقلان. ولكن، $Var(X + Y) = Var(X) + Var(Y)$ لا يمكن أن تطبق إلا إذا كانا مستقلان.

التغاير

التغاير لمتغيرين عشوائيين X و Y نعرفها كالآتي:

$$Cov(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

مرة أخرى، يمكننا القيام ببعض العمليات الرياضية ونحصل على التالي:

$$Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

لاحظ أنه على الرغم أن التباين للمتغير العشوائي يجب أن يكون رقم إيجابي، التغاير لمتغيرين عشوائيين يمكن أن يكون سلبي. بالأصح، التغاير يساعد على قياس العلاقة بين متغيرين؛ علامة نتيجة حساب التغاير تساعدنا على معرفة العلاقة بين المتغيرين إذا كانت إيجابية أو سلبية. إذا كانا المتغيرين X و Y مستقلين، فإن $Cov(X, Y) = 0$ و $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$

متغيرات برنولي العشوائية

لنفترض أننا نستخدم المتغير العشوائي X لتمثيل عملية رمي العملة المعدنية بشكل منحاز وفيها $P(Heads) = p$. يمكننا القول إن $X = 1$ إذا كان رمي العملة يؤدي إلى صوره، و $X = 0$ إذا كان رمي العملة يؤدي إلى كتابه. لذا، $P(X = 1) = p$ و $P(X = 0) = 1 - p$. هذا النوع من المتغيرات العشوائية الثنائية يطلق عليه نافير برنولي العشوائي؛ يمكننا حساب القيمة المتوقعة والتبيان بالطريقة التالية: 📝

$$\mathbb{E}[X] = 1 \times p + 0 \times (1 - p) = p$$

$$
\begin{aligned}
Var(X) &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\
&= 1^2 \times p + 0^2 \times (1-p) - p^2 \\
&= p - p^2 \\
&= p(1-p)
\end{aligned}
$$

متوسط العينة

لنفترض أن لدينا عملة معدنية مُنحازة وفيها $P(Heads) = p$ ونريد أن نتوقع p. يمكننا رمي العملة n مرات وأخذ عينة من الرميات وحساب نسبة ظهور الصورة في عينتينا، نسميها $\hat{p}$. إذا عرفنا أن $\hat{p}$ أقرب بشكل أكثر أن تكون p (صوره)، يمكننا استخدام $\hat{p}$ كوسيلة توقع أو تقدير ل p.

لاحظ أن p ليست قيمه عشوائية؛ هي قيمه ثابتة بناءًا على الانحياز في عملتنا المعدنية. $\hat{p}$ هي قيمه عشوائية كونها تتكون من الرمي العشوائي للعملة المعدنية. لذا، يمكننا حساب التوقع والتباين ل $\hat{p}$ لنعرف مدى جودة تقديره ل p.

لحساب $\mathbb{E}[\hat{p}]$، سنحتاج أولاً لتعريف المتغيرات العشوائية لكل عملية رمي للعملة المعدنية في عينتنا. لتكون X_i هي متغير برنولي العشوائي للقيمة i^{th} من رمي العملة. إذا، نعرف أن:

$$
\hat{p} = \frac{X_1 + X_2 + \ldots + X_n}{n}
$$

لحساب التوقع ل $\hat{p}$، يمكننا تعويض المعادلة في الأعلى ونستخدم ما نعرفه أن $\mathbb{E}[X_i] = p$ بما أن X_i هي متغير برنولي العشوائي.

$$
\begin{aligned}
\mathbb{E}[\hat{p}] &= \mathbb{E}\left[\frac{X_1 + X_2 + \ldots + X_n}{n}\right] \\
&= \frac{1}{n}\mathbb{E}[X_1 + \ldots + X_n] \\
&= \frac{1}{n}(\mathbb{E}[X_1] + \ldots + \mathbb{E}[X_n]) \\
&= \frac{1}{n}(p + \ldots + p) \\
&= \frac{1}{n}(np) \\
\mathbb{E}[\hat{p}] &= p
\end{aligned}
$$

وجدنا أن $\mathbb{E}[\hat{p}] = p$. بمعنى آخر، بعد كافي من عمليات رمي للعملة نتوقع أن $\hat{p}$ ستتلاق مع الانحياز الحقيقي في العملة p. نقول إن $\hat{p}$ هي **متوقع (مقدر) غير متحيز Unbiased Estimator** ل p.

الآن، نقوم بحساب التباين ل $\hat{p}$. بما أن كل عملية رمي للعملة هي عملية مستقلة عن باقي الرميات، فأننا نعرف أن X_i هي مستقلة. ذلك يُمكننا من استخدام خطية التباين:

$$
\begin{aligned}
Var(\hat{p}) &= Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) \\
&= \frac{1}{n^2} \times np(1-p) \\
Var(\hat{p}) &= \frac{p(1-p)}{n}
\end{aligned}
$$

من المعادلة السابقة، نرى أن أداة التوقع لدينا تباينها يقل كلما زادت n، عدد مرات الرمي في عينتنا. بمعنى آخر، إذا قمنا بجمع الكثير من البيانات فسنكون متأكدين من نتائج أداة توقعنا. يعرف ذلك بقاعدة الأرقام الكبيرة. 📝📝

ملخص التوقع والتباين

استخدمنا التوقع والتباين لشرح بشكل بسيط متوسط وانتشار العينة العشوائية. هذه الأدوات الرياضية تساعدنا على تحديد مدى كفاءة قيمه محسوبة من العينة على توقع باقي المجتمع الإحصائي.

التقليل من دالة الخسارة يكون لنا نموذجاً دقيقاً على بيانات التدريب. التوقع والتباين يساعدنا على إظهار بيانات عن دقة النموذج عندما يرى بيانات جديده من المجتمع الإحصائي.

المخاطر

في مثال لأحد النماذج التي تحدثنا عنها في الفصول السابقة، النادل قام بجمع بيانات الإكراميات التي حصل عليها في إحدى الشهور. قمنا باختيار نموذج وتقليل دالة الخسارة للخطأ التربيعي المتوسط (MSE)، لنتأكد أن نموذجنا يقدم نتائج أفضل من بقية النماذج الأخرى على هذه البيانات. لهذا النموذج متغير واحد فقط وهو θ. وجدنا أيضاً أن متغير التحسين لدالة الخسارة MSE هو $\hat{\theta} = \text{mean}(\mathbf{y})$.

على الرغم أن مثل هذه النماذج تقدم توقعات وتنبؤات صحيحه على بيانات التدريب، نريد أن نعرف ما إذا كان النموذج يستطيع القيام بنفس الأداء على بيانات جديده من المجتمع الإحصائي للبدء في التعامل مع هذه المشكلة، سنتعرف على المصطلح الإحصائي **المخاطر Risk**، وتعرف أيضاً ب **الخسارة المتوقعة Expected Loss**. 📝

التعريف

مخاطر النموذج هي الخسارة المتوقعه للنموذج عند تجربته على قيمه عشوائية من المجتمع الأحصائي.

في مثالنا، المجتمع الإحصائي هو جميع الإكراميات التي حصل عليها النادل طوال فترة عمله، بما فيها الإكراميات المستقبليه. نستخدم المتغير العشوائي X لتمثيل نسبة الإكرامية العشوائية المختاره من المجتمع، والمتغير θ يمثل توقع النموذج. بإستخدام هذه الرموز، فأن المخاطر $R(\theta)$ لنموذجنا هي:

$$R(\theta) = \mathbb{E}\left[(X-\theta)^2\right]$$

في المعادلة، استخدمنا دالة الخسارة ل MSE والتي تعطينا $(X-\theta)^2$. المخاطر هي دالة θ كوننا نتحكم بمحتوى θ كما يحلو لنا.

على عكس الخسارة، استخدام المخاطر يسمح لنا بتحديد أسباب دقة النموذج على المجتمع بشكل عام. إذا كانت نتيجة المخاطر لنموذجنا قليله، فأنه سيقوم بتنبؤات صحيحه على بيانات المجتمع على المدى البعيد. في الجانب الآخر، إذا كانت المخاطر لنموذجنا عاليه فإن أداءه على بيانات المجتمع سيكون ضعيفاً.

بالطبع، نحاول أن نختار قيمه ل θ تجعل مخاطر النموذج أقل ما يمكن. نستخدم المتغير θ^* لتمثيل القيمة الناتجة عن استخدام θ للحصول على أقل مخاطر، أو أفضل متغير للحصول على نموذج مثالي للبيانات. للتوضيح، θ^* تمثل النموذج الذي يقلل المخاطر بينما $\hat{\theta}$ تمثل المتغير الذي يقلل من دالة الخسارة للبيانات.

تقليل المخاطر

لنحاول إيجاد قيمة θ التي تقلل من المخاطر. في السابق، كنا نستخدم التفاضل والتكامل للقيام بعملية التقليل هذه. الآن، سنستخدم أدوات رياضية ذا مصطلحات واضحة. سنبدل $X-\theta$ ب $X-\mathbb{E}[X]+\mathbb{E}[X]-\theta$ ونتوسع:

$$\begin{aligned}
R(\theta) &= \mathbb{E}[(X-\theta)^2] \\
&= \mathbb{E}\left[(X-\mathbb{E}[X]+\mathbb{E}[X]-\theta)^2\right] \\
&= \mathbb{E}\left[\big((X-\mathbb{E}[X])+(\mathbb{E}[X]-\theta)\big)^2\right] \\
&= \mathbb{E}\left[(X-\mathbb{E}[X])^2+2(X-\mathbb{E}[X])(\mathbb{E}[X]-\theta)+(\mathbb{E}[X]-\theta)^2\right]
\end{aligned}$$

الآن، نطبق خطية التوقع ونبسط المعادلة. سنستخدم $\mathbb{E}\left[(X-\mathbb{E}[X])\right]=0$ وهي تعني بشكل كبير أن $\mathbb{E}[X]$ تقع في متوسط التوزيع ل X.

$$\begin{aligned}
R(\theta) &= \mathbb{E}\left[(X-\mathbb{E}[X])^2\right]+\mathbb{E}\left[2(X-\mathbb{E}[X])(\mathbb{E}[X]-\theta)\right]+\mathbb{E}\left[(\mathbb{E}[X]-\theta)^2\right] \\
&= \mathbb{E}\left[(X-\mathbb{E}[X])^2\right]+2(\mathbb{E}[X]-\theta)\underbrace{\mathbb{E}\left[(X-\mathbb{E}[X])\right]}_{=0}+(\mathbb{E}[X]-\theta)^2 \\
&= \mathbb{E}\left[(X-\mathbb{E}[X])^2\right]+0+(\mathbb{E}[X]-\theta)^2 \\
R(\theta) &= \mathbb{E}\left[(X-\mathbb{E}[X])^2\right]+(\mathbb{E}[X]-\theta)^2
\end{aligned}$$

لاحظ أن أول مصطلح في المعادلة السابقة هو **التباين Variance** ل X، $Var(X)$، والذي لا يعتمد على θ. المصطلح الثاني يعطي مقياس لمدى قرب θ من $\mathbb{E}[X]$. بسبب ذلك، يطلق على المصطلح الثاني **الانحياز Bias** للنموذج. بمعنى آخر، مخاطر النموذج هي انحياز النموذج إضافة إلى التبيان للكمية التي نرغب بالتنبؤ عنها:

$$R(\theta) = \underbrace{(\mathbb{E}[X]-\theta)^2}_{\text{bias}}+\underbrace{Var(X)}_{\text{variance}}$$

ولذلك، تكون المخاطر أقل عندما لا يكون نموذجنا مُنحاز: $\theta^* = \mathbb{E}[X]$

تحليل المخاطر

لاحظ عندما يكون نموذجنا دون تحيز، المخاطر تكون إيجابية. ذلك يعني أنه حتى النموذج المثالي سيحصل على توقعات خاطئة. ذلك يحدث لأن النموذج سيتنبأ بقيمه واحدة فقط بينما X قد تأخذ أي قيمه من المجتمع الإحصائي. التباين يبين لنا حجم وكمية الخطأ. إذا كان التباين قليل يعني أن X ستكون قيمه قريبه من θ، وعندما يكون التباين عالي فيعني أن X ستكون قيمه بعيدة عن θ.

تقليل المخاطر التجريبية

من التحليل السابق، نريد تعيين $\theta = \mathbb{E}[X]$. للأسف، لحساب $\mathbb{E}[X]$ نحتاج لمعرفة كل شيء عن المجتمع الإحصائي. لمعرفة السبب، تحقق من التعبير التالي ل $\mathbb{E}[X]$:

$$\mathbb{E}[X] = \sum_{x\in\mathbb{X}} x \cdot P(X=x)$$

تمثل $P(X = x)$ احتمالية أن X تأخذ قيمه معينة من المجتمع الإحصائي. لكن، لحساب الاحتمالية، نحتاج لمعرفة جميع القيم التي يمكن أن تأخذها X وعدد مرات ظهورها في المجتمع الإحصائي. بمعنى آخر، لتقليل المخاطر في النموذج بشكل مُتقن، نحتاج أن تتوفر لدينا بيانات المجتمع الإحصائي

يمكننا مواجهة هذه المشكلة عن طريق معرفة أن توزيع القيم في عينة عشوائية كبيرة سيكون قريباً من توزيع القيم في المجتمع الإحصائي. إذا كان لذلك صحيحاً في عينتنا، يمكننا التعامل مع العينة كأنها كامل المجتمع الإحصائي.

لنفترض أننا رسمنا نقاطاً بشكل عشوائي من العينة بدلاً من المجتمع الإحصائي. بما أن لدينا عدد n من النقاط في العينة $\mathbf{x} = x_1, x_2, \ldots, x_n$، كل نقطة x_i لديها احتمالية تساوي $\frac{1}{n}$ للظهور. الآن يمكننا كتابة توقع ل $\mathbb{E}[X]$:

$$\mathbb{E}[X] \approx \frac{1}{n} \sum_{i=1}^{n} x_i = \text{mean}(\mathbf{x})$$

لذا، توقعنا المناسب ل θ^* باستخدام المعلومات من العينة العشوائية هو أن $\hat{\theta} = \text{mean}(\mathbf{x})$. نقول إن $\hat{\theta}$ تقلل من **المخاطر التجريبية Empirical Risk**، وهي المخاطر التي تم حسابها باستخدام العينة كبديل عن المجتمع الإحصائي.

أهمية العينات العشوائية

يجب أن نلاحظ أهمية العينة العشوائية في المثال السابق. إذا كانت العينة غير عشوائية، لن نستطيع إثبات الفرضية السابقة أن توزيع العينة يشابه توزيع المجتمع الإحصائي. باستخدام عينه غير عشوائية لتوقع θ^* ستكون النتائج عادةً منحازة وذات مخاطر أعلى.

العلاقة مع تقليل الخسارة

لنتذكر أننا شرحنا مسبقاً بأن $\hat{\theta} = \text{mean}(\mathbf{x})$ تقلل من دالة الخسارة MSE في البيانات. الآن، أخذنا خطوه جادة إلى الأمام. إذا كانت بيانات التدريب هي عينه عشوائية، فإن $\hat{\theta} = \text{mean}(\mathbf{x})$ لن تكون لنا النموذج المثالي فقط لبيانات التدريب ولكن أيضاً لكامل المجتمع الإحصائي، بالأخذ بالاعتبار المعلومات التي لدينا عن العينة.

ملخص المخاطر

باستخدام الأدوات الرياضية التي طورناها في هذا الفصل، تعرفنا على أداء النموذج على المجتمع الإحصائي. النموذج يكون ذو تنبؤات دقيقة إذا كانت **المخاطر الإحصائية Statistical Risk** أقل. وجدنا أن المتغير للنموذج المثالي هو:

$$\theta^* = \mathbb{E}[X]$$

بما أننا لا نستطيع القيام بعملية الحساب بسهوله، وجدنا متغير النموذج الذي يقلل من **المخاطر التجريبية Empirical Risk**:

$$\hat{\theta} = \text{mean}(\mathbf{x})$$

إذا كانت بيانات التدريب أخذت بشكل عشوائي من المجتمع الإحصائي، فغالباً $\hat{\theta} \approx \theta^*$. لذا، النموذج الذي يتم تدريبه باستخدام بيانات عشوائية ضخمة من المجتمع الإحصائي غالباً ما يؤدي بشكل ممتاز على كامل المجتمع الإحصائي.

النماذج الخطية

مقدمة

الآن، بعد أن تعلمنا بشكل عام أدوات لضبط النماذج مع دوال التكلفة، نتحول لطرق تحسين النموذج. للتبسيط، في السابق حددنا عملنا على النموذج الثابت: نموذجنا فقط يتوقع رقم واحد.

ولكن، إعطاء نموذج كهذا للنادل لن يرضيه. يود النادل أن يوضح أنه حصل على معلومات أكثر نسبة الإكرامية من الطاولات التي خدمها. لماذا لا نستخدم بياناته الأخرى، مثلاً حجم العملاء في الطاولة ومجموع الفاتورة، لغرض جعل النموذج أكثر فائدة.

في هذا الفصل سنتعرف على النماذج الخطية والذي يسمح لنا باستخدام جميع البيانات التي لدينا لإجراء توقعات. النماذج الخطية لا تستخدم بنطاقٍ واسع فقط، بل أيضاً لديها أسس نظرية غنية بالمعلومات التي تجعلنا نفهم أدوات مستقبليه للنماذج. سنتعرف على نموذج الانحدار الخطي البسيط الذي يستخدم متغير واحد، سنتعلم كيف يستخدم النزول الاشتقاقي لضبط النموذج، وأخيراً التوسع في النموذج وإضافة المزيد من المتغيرات له.

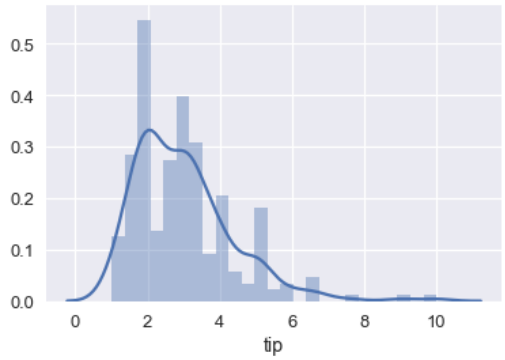
التنبؤ بالإكراميات

سابقاً، تعاملنا مع بيانات تحتوي على صف واحد لكل طاولة قام النادل بخدمتها لمدة أسبوع. النادل قام بجمع البيانات للقيام بالتوقع بقيمة الإكرامية التي سيحصل عليها في المستقبل:

```
tips = sns.load_dataset('tips')
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.5	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
sns.distplot(tips['tip'], bins=25);
```



كما تحدثنا سابقاً، إذا اخترنا النموذج الثابت ودالة الخطأ التربيعي المتوسط، فإن نموذجنا سيتوقع متوسط الإكراميات:

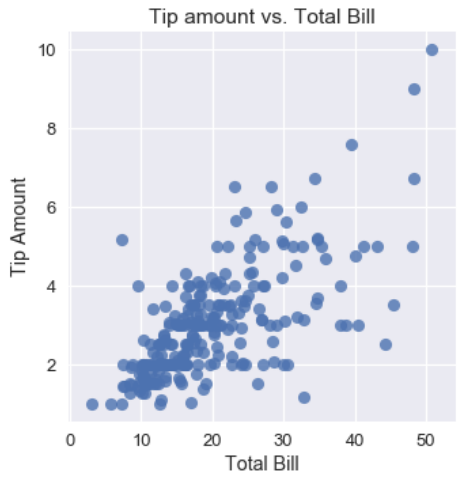
```
np.mean(tips['tip'])
```

```
2.9982786885245902
```

يعني ذلك أن عندما يأتي مجموعة من الزبائن للنادل ثم يسألنا النادل عن مجموعه الإكرامية التي سيحصل عليها، فسنجيب عليه "حوالي 3$"، أياً كان عدد الزبائن ومجموع الفاتورة.

ولكن، عند التحقق من باقي المتغيرات في البيانات، يمكننا أن نقوم بتوقعات دقيقه إذا أضفنا تلك المتغيرات للنموذج. مثلاً، الرسم البياني التالي يوضح مجموع الإكرامية مقارنة بمبلغ الفاتورة ويظهر العلاقة الإيجابية بينهما:

```
sns.lmplot(x='total_bill', y='tip', data=tips, fit_reg=False)
plt.title('Tip amount vs. Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Tip Amount');
```



على الرغم من أن متوسط الإكرامية هو $3، إذا كانت الطاولة طلبت ما مجموعه $40 من الوجبات فأننا متأكدين أن النادل سيحصل على أكثر من $3 كإكرامية. لذا، نريد التعديل في نموذجنا ليتمكن من التوقع بناءًا على متغيرات في بياناتنا بدلاً من توقع متوسط الإكرامية. للقيام بذلك، سنستخدم النموذج الخطي بدلاً من الثابت.

لنراجع أولاً الأدوات التي لدينا لبناء النموذج والتوقع وتعريف بعض الرموز الجديدة لنتمكن بشكل أفضل من تمثيل العمليات الرياضية الإضافية في النموذج الخطي.

تعريف نموذج خطي بسيط

نحن مهتمون بتوقع قيمة الإكرامية بناءًا على مجموع الفاتورة. لنجعل y تمثل مجموع الإكرامية، المتغير الذي نريد أن يتوقعه النموذج. و x تمثل مجموع الفاتورة، المتغير الذي نريد استخدامه للتوقع.

نقوم بتعريف النموذج الخطي $f^*_{\boldsymbol{\theta}}$ الذي يعتمد على x:

$$f^*_{\boldsymbol{\theta}}(x) = \theta^*_1 x + \theta^*_0$$

نتعامل مع $f^*_{\boldsymbol{\theta}}(x)$ على أنها الدالة التي أنشأت البيانات.

تفترض $f^*_{\boldsymbol{\theta}}(x)$ أن y لديها علاقة خطيه كاملة مع x. ولكن، في بياناتنا لا يظهر لنا خط مستقيم بسبب وجود بعض البيانات العشوائية المزعجة ϵ. رياضياً، نأخذ بعين الاعتبار هذه المشكلة بإضافة المصطلح:

$$y = f^*_{\boldsymbol{\theta}}(x) + \epsilon$$

إذا كانت الافتراضية أن y لديها علاقه خطية كاملة مع x، وتمكنا بطريقة ما من توقع القيمة الصحيحة ل θ^*_1 و θ^*_0، وبشكل غير اعتيادي لم يكنا لدينا أي عشوائية في البيانات، سنتمكن من التوقع بشكل دقيق قيمة الإكرامية التي سيحصل عليها النادل من جميع الزبائن، وبشكل دائم. بالطبع، لا يمكننا تلبية جميع المعايير في الواقع. بدلاً من ذلك، نتوقع θ^*_1 و θ^*_0 باستخدام البيانات لجعل توقعاتنا أقرب دقه للواقع.

توقع النموذج الخطي

بما أننا لا نستطيع إيجاد قيمة θ^*_1 و θ^*_0 بشكل دقيق، سنفترض أن بياناتنا تتوقع قيمة هذه المتغيرات. نشير للتوقعات ب θ_1 و θ_0، وتوقعنا الذي يتم ضبطه بالنموذج ب $\hat{\theta_1}$ و $\hat{\theta_0}$، و النموذج:

$$f_{\boldsymbol{\theta}}(x) = \theta_1 x + \theta_0$$

أحياناً سترى $h(x)$ بدلاً من $f_{\hat{\theta}}(x)$؛ ال h تعني الفرضية *Hypothesis*، كون $f_{\hat{\theta}}(x)$ هي فرضيتنا ل $f_{\theta^*}(x)$.

من أجل تحديد قيمة $\hat{\theta_1}$ و $\hat{\theta_0}$، نقوم باختيار دالة تكلفة وتقليلها باستخدام النزول الاشتقاقي.

ضبط النموذج الخطي باستخدام النزول الاشتقاقي

نريد ضبط نموذج خطي يتنبأ بقيمة الإكرامية من مجموع الفاتورة:

$$f_{\boldsymbol{\theta}}(x) = \theta_1 x + \theta_0$$

ولتحسين قيم θ_1 و θ_0، نحتاج أولاً لاختيار دالة خسارة. سنختار دالة خسارة الخطأ التربيعي المتوسط MSE:

$$L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_{\boldsymbol{\theta}}(x_i))^2$$

لاحظ أننا عدلنا على دالة الخسارة لتوضيح إضافتنا لمتغير في النموذج. الآن، $\mathbf{x}$ هي مصفوفة أحادية البعد تحتوي على جميع الفواتير، و $\mathbf{y}$ هي مصفوفة أحادية البعد تحتوي على قيمة كل إكرامية، و $\boldsymbol{\theta}$ هي مصفوفة تحتوي على التالي: $\boldsymbol{\theta} = [\theta_1, \theta_0]$.

يطلق على استخدام النموذج الخطي مع دالة خسارة الخطأ التربيعي المتوسط بإسم الانحدار الخطي للمربعات الصغرى Least-squares Linear Regression. يمكننا استخدام النزول الاشتقاقي لإيجاد قيمة $\boldsymbol{\theta}$ التي تقلل الخسارة. 📝

ملاحظة عن استخدام العلاقات

إذا سبق أن رأيت الانحدار الخطي للمربعات الصغرى، قد تلاحظ أننا نستطيع حساب معامل الارتباط واستخدامه لتحديد قيمة θ_1 و θ_0. هذه طريقة أسهل وأسرع للحساب بدلاً من استخدام النزول الاشتقاقي في أي معادله، تماماً كما يكون أسهل لنا حساب المتوسط بدلاً من حساب النزول الاشتقاقي لضبط النموذج الثابت. على أية حال، سنستخدم النزول الاشتقاقي لأنها طريقة عامله لتقليل الخسارة وستعمل معنا لاحقاً عندما نتعرف على نماذج لا يمكن حساب خسارتها إحصائياً. بالأصح، في كثير من المشاكل في العالم الحقيقي، سنستخدم النزول الاشتقاقي حتى ولو كانت هناك طرق إحصائية تحليله لأن حسابها يأخذ وقتاً أطول من النزول الاشتقاقي، خاصة عندما تكون البيانات ذات حجم كبير.

مشتقة خسارة الخطأ التربيعي المتوسط MSE

لاستخدام النزول الاشتقاقي، نحتاج لحساب مشتقة خسارة الخطأ التربيعي المتوسط بالنسبة ل $\boldsymbol{\theta}$. الآن بما أن $\boldsymbol{\theta}$ عبارة عن مصفوفة ذات طول 2 وليست قيمة عددية مدرجة Scalar، و $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y})$ أيضاً مصفوفة من الحجم 2. 📝

$$\begin{aligned}\nabla_{\theta} L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) &= \nabla_{\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(x_i))^2 \right] \\ &= \frac{1}{n} \sum_{i=1}^{n} 2(y_i - f_{\boldsymbol{\theta}}(x_i))(-\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i)) \\ &= -\frac{2}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(x_i))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i))\end{aligned}$$

نعرف أن:

$$f_{\boldsymbol{\theta}}(x) = \theta_1 x + \theta_0$$

نريد حساب قيمة $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i)$ والتي هي مصفوفة طولها 2:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i) &= \begin{bmatrix} \frac{\partial}{\partial \theta_0} f_{\boldsymbol{\theta}}(x_i) \\ \frac{\partial}{\partial \theta_1} f_{\boldsymbol{\theta}}(x_i) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial}{\partial \theta_0} [\theta_1 x_i + \theta_0] \\ \frac{\partial}{\partial \theta_1} [\theta_1 x_i + \theta_0] \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ x_i \end{bmatrix}\end{aligned}$$

أخيراً، نعوضها في معادلتنا الأساسية لنحصل على التالي:

$$\begin{aligned}\nabla_{\theta} L(\theta, \mathbf{x}, \mathbf{y}) &= -\frac{2}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(x_i))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i)) \\ &= -\frac{2}{n} \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(x_i)) \begin{bmatrix} 1 \\ x_i \end{bmatrix} \\ &= -\frac{2}{n} \sum_{i=1}^{n} \begin{bmatrix} (y_i - f_{\boldsymbol{\theta}}(x_i)) \\ (y_i - f_{\boldsymbol{\theta}}(x_i)) x_i \end{bmatrix}\end{aligned}$$

هذه مصفوفة من طولها 2 كون $(y_i - f_{\boldsymbol{\theta}}(x_i))$ قيمة عددية مدرجه.

تطبيق النزول الاشتقاقي

الآن، لنقوم بضبط النموذج الخطي على بيانات الإكراميات لتوقع قيمة الإكرامية من مجموع الفاتورة.

أولاً، نقوم بتعريف دالة في بايثون لحساب الخسارة:

```
def simple_linear_model(thetas, x_vals):
    '''نتيجة هذه الداله هي القيمه المتوقعه من النموذج الخطي'''
    return thetas[0] + thetas[1] * x_vals

def mse_loss(thetas, x_vals, y_vals):
    return np.mean((y_vals - simple_linear_model(thetas, x_vals)) ** 2)
```

ثم نعرف دالة تقوم بحساب خطية الخسارة:

```
def grad_mse_loss(thetas, x_vals, y_vals):
    n = len(x_vals)
    grad_0 = y_vals - simple_linear_model(thetas, x_vals)
    grad_1 = (y_vals - simple_linear_model(thetas, x_vals)) * x_vals
    return -2 / n * np.array([np.sum(grad_0), np.sum(grad_1)])
```

سنقوم باستخدام الدالة `minimize` التي سبق أن عرفناها لتطبيق النزول الاشتقاقي:

```
def minimize(loss_fn, grad_loss_fn, x_vals, y_vals,
             alpha=0.0005, progress=True):
    # تستخدم النزول الاشتقاقي للتقليل من دالة الخسارة loss_fn.
    # عندما يكون theta_hat (θ^) تنتج لنا الداله القيمه الصغرى ل
    # التغيير اقل من 0.001 بين التكرارات.
    theta = np.array([0., 0.])
    loss = loss_fn(theta, x_vals, y_vals)
    while True:
        if progress:
            print(f'theta: {theta} | loss: {loss}')
        gradient = grad_loss_fn(theta, x_vals, y_vals)
```

```
        new_theta = theta - alpha * gradient
        new_loss = loss_fn(new_theta, x_vals, y_vals)

        if abs(new_loss - loss) < 0.0001:
            return new_theta

        theta = new_theta
        loss = new_loss
```

والآن نقوم بتطبيق النزول الاشتقاقي:

```
thetas = minimize(mse_loss, grad_mse_loss, tips['total_bill'], tips['tip'])
```

```
theta: [0. 0.] | cost: 10.896283606557377
theta: [0.   0.07] | cost: 3.8937622006094705
theta: [0.  0.1] | cost: 1.9359443267168215
theta: [0.01 0.12] | cost: 1.388538448286097
theta: [0.01 0.13] | cost: 1.235459416905535
theta: [0.01 0.14] | cost: 1.1926273731479433
theta: [0.01 0.14] | cost: 1.1806184944517062
theta: [0.01 0.14] | cost: 1.177227251696266
theta: [0.01 0.14] | cost: 1.1762453624313751
theta: [0.01 0.14] | cost: 1.1759370980989148
theta: [0.01 0.14] | cost: 1.175817178966766
```

نلاحظ أن النزول الاشتقاقي يقترب لقيمة $\hat{\theta}_0 = 0.01$ و $\hat{\theta}_0 = 0.14$. نموذجنا الخطي الآن:

$$y = 0.14x + 0.01$$

يمكننا استخدام النتيجة السابقة لرسم توقعاتنا بجانب البيانات الحقيقية:

```
x_vals = np.array([0, 55])
sns.lmplot(x='total_bill', y='tip', data=tips, fit_reg=False)
plt.plot(x_vals, simple_linear_model(thetas, x_vals), c='goldenrod')
plt.title('Tip amount vs. Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Tip Amount');
```

نلاحظ أنه عندما تكون قيمة الفاتورة $10، فإن نموذجنا يتوقع أن النادل سيحصل على إكرامية بحوالي $1.50. بنفس الطريقة، إذا كانت قيمة الفاتورة $40 فإن النموذج يتوقع حصول النادل على إكرامية بقيمة $6.00.

الانحدار الخطي المتعدد

نموذجنا الخطي البسيط لميزة إضافية عن النموذج الثابت، الميزة هي استخدامه البيانات للتوقع. ولكن، لا يزال النموذج محدود كونه يستخدم متغير واحد من بياناتنا. الكثير من البيانات تحتوي على أكثر من متغير مهم ومفيد للاستخدام، ويمكن للانحدار الخطي المتعدد الاستفادة من ذلك. مثلاً، لنأخذ البيانات التالية لأنواع السيارات ومعلومات صرف الوقود بالميل لكل جالون (Milage Per Gallon MPG):

لتحميل قاعدة البيانات mpg.csv اضغط هنا.

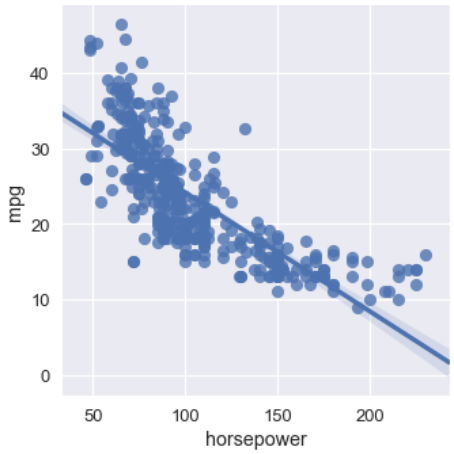
```
mpg = pd.read_csv('mpg.csv').dropna().reset_index(drop=True)
mpg
```

	mpg	cylinders	displacement	...	model year	origin	car name
0	18	8	307	...	70	1	chevrolet chevelle malibu
1	15	8	350	...	70	1	buick skylark 320
2	18	8	318	...	70	1	plymouth satellite
...	...	...	...	...	...	...	...
389	32	4	135	...	82	1	dodge rampage
390	28	4	120	...	82	1	ford ranger
391	31	4	119	...	82	1	chevy s-10

```
392 rows × 9 columns
```

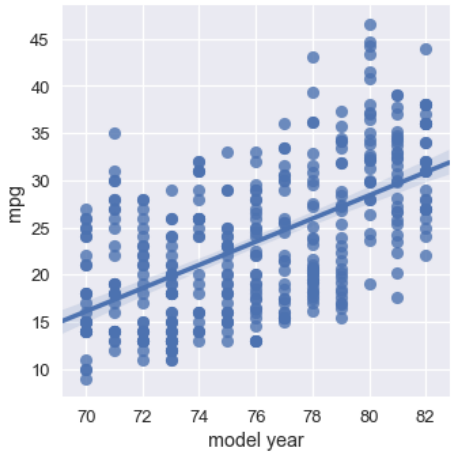
يبدو لنا أن أكثر من متغير يؤثر على صرف السيارة للوقود. مثلاً، يبدو أن صرف الوقود يقل عندما تزيد قوة الحصان للسيارة:

```
sns.lmplot(x='horsepower', y='mpg', data=mpg);
```



ولكن، السيارات التي في السنوات الأخيرة لديها صرف وقود أفضل بشكل عام عن السيارات القديمة:

```
sns.lmplot(x='model year', y='mpg', data=mpg);
```



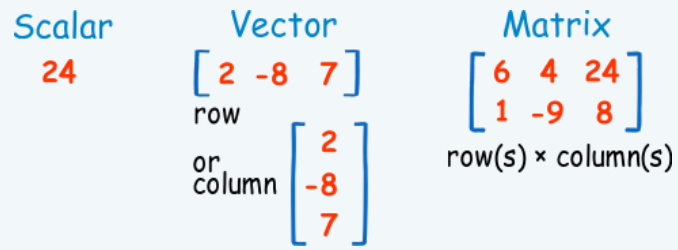
يظهر أن بإمكاننا الحصول على نتائج أكثر دقة للنموذج إذا استطعنا استخدام قوة الحصان وسنة صناعة السيارة للتنبؤ عن كمية صرف الوقود بالميل لكل جالون MPG. بالأصح، يبدو أن النموذج المثالي يأخذ بعين الاعتبار جميع المتغيرات الرقمية في بياناتنا. يمكننا توسيع نموذجنا الخطي ذو المتغير الواحد ليتمكن من التنبؤ بناءًا على أي عدد من المتغيرات.

ذكرنا أن تعريف النموذج كالتالي:

$$f_{\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p$$

فيها $\mathbf{x}$ تمثل متّجهة Vector تحتوي على عدد p من المتغيرات لسيارة واحدة. النموذج السابق يقول التالي، "خذ أكثر من متغير عن السيارة، اضربهم بوزن Weight معين، ثم أجمعهم معاً للقيام بتوقع صرفية السيارة للوقود بالميل لكل جالون".

توجد أنواع متعددة من طرق تمثيل الأرقام:



- عدد Scalar: رقم صحيح مثلاً $7, -4, 0.345$.
- المتّجه Vector: هي مصفوفة أرقام أحادية الأبعاد، تكون إما من صف واحد أو عمود واحد.
- مصفوفة Matrix: مصفوفة أرقام تحتوي على أكثر من صف أو عمود.

مثلاً، إذا أردنا إجراء توقع لأول سيارة في بياناتنا باستخدام قوة الحصان، الوزن، وسنة الصناعة، فسيكون شكل المتّجهة $\mathbf{x}$ كالتالي:

```
mpg.loc[0:0, ['horsepower', 'weight', 'model year']]
```

	horsepower	weight	model year
0	130.0	3504.0	70

في هذا المثال أبقينا أسماء الأعمدة للتوضيح، ولكن تذكر أن $\mathbf{x}$ تحتوي فقط على القيم الرقمية من الجدول السابق: $\mathbf{x} = [130.0, 3504.0, 70]$.

الآن، سنقوم بتعريف طريقه حسابية ستسهل العمليات الحسابية القادمة. سنقوم بإضافة الرقم 1 إلى المتّجهة $\mathbf{x}$، وسيكون شكل الصف كالتالي:

```
mpg_mat = mpg.assign(bias=1)
mpg_mat.loc[0:0, ['bias', 'horsepower', 'weight', 'model year']]
```

	bias	horsepower	weight	model year
0	1	130.0	3504.0	70

الآن، لاحظ ما سيحدث للعملية الحسابية لنموذجنا:

$$\begin{aligned} f_{\theta}(\mathbf{x}) &= \theta_0 + \theta_1 x_1 + \ldots + \theta_p x_p \\ &= \theta_0(1) + \theta_1 x_1 + \ldots + \theta_p x_p \\ &= \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_p x_p \\ f_{\theta}(\mathbf{x}) &= \boldsymbol{\theta} \cdot \mathbf{x} \end{aligned}$$

فيها $\boldsymbol{\theta} \cdot \mathbf{x}$ هي متّجه لحاصل ضرب $\boldsymbol{\theta}$ و $\mathbf{x}$. صُممت المتّجهات والمصفوفات لكتابة التركيبات الخطية ولذلك فهي مناسبة جداً لنموذجنا الخطي. ولكن، يجب عليك من الآن وصاعدًا تذكر أن $\boldsymbol{\theta} \cdot \mathbf{x}$ هي حاصل ضرب متّجه بأخرى. يمكن أيضاً لنزع الشك، توسيع عملية ضرب المتجهتين إلى عملية جمع وضرب مبسطة.

الآن، نقوم بتعريف المصفوفة $\mathbf{X}$ والتي ستكون المصفوفة التي تحتوي على جميع أنواع السيارات كصفوف، وأول عمود هو قيمة التحيز Bias. مثلاً، هذه أول خمس أسطر من المصفوفة $\mathbf{X}$:

```
mpg_mat = mpg.assign(bias=1)
mpg_mat.loc[0:4, ['bias', 'horsepower', 'weight', 'model year']]
```

	bias	horsepower	weight	model year
0	1	130.0	3504.0	70
1	1	165.0	3693.0	70
2	1	150.0	3436.0	70
3	1	150.0	3433.0	70
4	1	140.0	3449.0	70

للتذكير مرة أخرى، المصفوفة الحقيقية $\mathbf{X}$ فقط تحتوي على القيم الرقمية من الجدول السابق.

لاحظ أن $\mathbf{X}$ تحتوي على أكثر من مُتجهة $\mathbf{x}$ فوق بعضها البعض. ليكون الوصف واضحاً، نقوم بتعريف $\mathbf{X}_i$ والتي ترمز للمتّجه في الصف رقم i في المصفوفة $\mathbf{X}$. نقوم بتعريف $X_{i,j}$ والتي تمثل القيمة ذات الرقم j في الصف ذو الرقم i في المصفوفة $\mathbf{X}$. لذا، $\mathbf{X}_i$ هي متّجه ذات أبعاد p و $\mathbf{X}$ هي مصفوفة $n \times p$، فيها n هي عدد السيارات لدينا و p هي عدد المتغيرات لكل سيارة. ددعيه {textbf{X}_{i,j\.

مثلاً، في الجدول السابق لدينا $\mathbf{X}_4 = [1, 140, 3449, 70]$ و $X_{4,1} = 140$ لذا الرموز هي مهمة عند تعريف دوال الخسارة لأننا سنحتاج إلى كلا القيمتين $\mathbf{X}$، مصفوفة البيانات المدخلة للنموذج، و y، متّجه صرف الوقود بالميل لكل جالون.

- $\mathbf{X}$ هي مصفوفة Matrix.
- $\mathbf{x}$ هي متّجه Vector وهي هنا كل صف على حدة. مثلاً السطر الثاني: $[1, 165.0, 3693.0, 70]$.
- j هي رقم صحيح Scalar مثلاً وزن السيارة في الصف الثالث 3436.0.

خسارة الخطأ التربيعي المتوسط وانحدارها

دالة خسارة الخطأ التربيعي المتوسط تأخذ متّجه بوزن $\boldsymbol{\theta}$ و مدخلات على شكل مصفوفة $\mathbf{X}$، و متّجه لصرف الوقود بالميل لكل جالون لكل سيارة $\mathbf{y}$:

$$L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \frac{1}{n}\sum_i (y_i - f_{\boldsymbol{\theta}}(\mathbf{X}_i))^2$$

أوجدنا مسبقاً مشتقة دالة خسارة الخطأ التربيعي المتوسط بالنسبة ل $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = -\frac{2}{n}\sum_i (y_i - f_{\boldsymbol{\theta}}(\mathbf{X}_i))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{X}_i))$$

نعرف أيضاً أن:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

لنقوم بحساب $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})$. عملية الحساب أسهل من المتوقع لأن $\boldsymbol{\theta} \cdot \mathbf{x} = \theta_0 x_0 + \ldots + \theta_p x_p$ إذاً $\frac{\partial}{\partial \theta_0}(\boldsymbol{\theta} \cdot \mathbf{x}) = x_0$ و $\frac{\partial}{\partial \theta_1}(\boldsymbol{\theta} \cdot \mathbf{x}) = x_1$ إلى آخره:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}} [\boldsymbol{\theta} \cdot \mathbf{x}] \\
&= \begin{bmatrix} \frac{\partial}{\partial \theta_0}(\boldsymbol{\theta} \cdot \mathbf{x}) \\ \frac{\partial}{\partial \theta_1}(\boldsymbol{\theta} \cdot \mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial \theta_p}(\boldsymbol{\theta} \cdot \mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_p \end{bmatrix} \\
\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbf{x}
\end{aligned}$$

أخيراً، نقوم بإدخال النتيجة لحساب الخطية:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) &= -\frac{2}{n}\sum_i (y_i - f_{\boldsymbol{\theta}}(\mathbf{X}_i))(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{X}_i)) \\
&= -\frac{2}{n}\sum_i (y_i - \boldsymbol{\theta} \cdot \mathbf{X}_i)(\mathbf{X}_i)
\end{aligned}$$

تذكر أنه بما أن $y_i - \boldsymbol{\theta} \cdot \mathbf{X}_i$ هي عدد و $\mathbf{X}_i$ هي متّجه ذات p أبعاد، الخطية $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$ هي أيضاً متّجه ذات p أبعاد.

رأينا نفس هذا النوع من النتائج عندما قمنا بحساب خطية الانحدار الخطي ووجدنا أنها ثنائية الأبعاد بما أن $\boldsymbol{\theta}$ كانت ثنائية الأبعاد.

ضبط النموذج الخطي مع النزول الاشتقاقي

يمكننا الآن إدخال الخسارة ومشتقاتها إلى دالة النزول الاشتقاقي. كالعادة، سنقوم بتعريف النموذج، دالة الخسارة و دالة النزول الاشتقاقيمشتقتها في بايثون:

```
def linear_model(thetas, X):
    '''Returns predictions by a linear model on x_vals.'''
    return  X @ thetas

def mse_loss(thetas, X, y):
    return np.mean((y - linear_model(thetas, X)) ** 2)
```

```
def grad_mse_loss(thetas, X, y):
    n = len(X)
    return -2 / n * (X.T @ y  - X.T @  X @ thetas)
```

استخدم الكاتب الرمز @ وهي علامة ضرب بين المصفوفات في Numpy، لذا يحتاج أن تكون X و thetas هي مصفوفات في Numpy كي يعمل الرمز @.

الآن، ببساطه يمكننا إدخال دوالنا إلى النزول الاشتقاقي:

```
X = (mpg_mat
     .loc[:, ['bias', 'horsepower', 'weight', 'model year']]
     .to_numpy())
y = mpg_mat['mpg'].to_numpy()

thetas = minimize(mse_loss, grad_mse_loss, X, y)
print(f'theta: {thetas} | loss: {mse_loss(thetas, X, y):.2f}')
```

```
theta: [ 0.  0.  0.  0.] | cost: 610.47
theta: [ 0.    0.    0.01  0.  ] | cost: 178.95
theta: [ 0.01 -0.11 -0.    0.55] | cost: 15.78
theta: [ 0.01 -0.01 -0.01  0.58] | cost: 11.97
theta: [-4.   -0.01 -0.01  0.63] | cost: 11.81
theta: [-13.72  -0.    -0.01   0.75] | cost: 11.65
theta: [-13.72  -0.    -0.01   0.75] | cost: 11.65
```

استخدم الكاتب في الكود البرمجي السابق الدالة minimize وهي مختلفة قليلاً عن السابقة، أجري عليها التعديلات التالية:

```
from scipy.optimize import minimize as sci_min
def minimize(loss_fn, grad_loss_fn, X, y, progress=True):
    # تستخدم دالة من مكتبة scipy للتقليل من خسارة الداله loss_fun
    # باستخدام نموذج من النزول الاشتقاقي

   theta = np.zeros(X.shape[1])
   iters = 0

   def objective(theta):
       return loss_fn(theta, X, y)
   def gradient(theta):
       return grad_loss_fn(theta, X, y)
   def print_theta(theta):
       nonlocal iters
       if progress and iters % progress == 0:
           print(f'theta: {theta} | loss: {loss_fn(theta, X, y):.2f}')
       iters += 1

   print_theta(theta)
   return sci_min(
       objective, theta, method='BFGS', jac=gradient, callback=print_theta,
       tol=1e-7
   ).x
```

بناءًا على النزول الاشتقاقي، فإن نموذجنا الخطي هو كالتالي:

$$y = -13.72 - 0.01x_2 + 0.75x_3$$

الرسم البياني لتنبؤاتنا

كيف أدى نموذجنا؟ نلاحظ أن الخسارة قلت بشكل كبير (من 610 حتى 11.6). يمكننا طباعة نتائج توقع النموذج بجانب النتائج الحقيقية:

```
reordered = ['predicted_mpg', 'mpg', 'horsepower', 'weight', 'model year']
with_predictions = (
    mpg
    .assign(predicted_mpg=linear_model(thetas, X))
    .loc[:, reordered]
)
with_predictions
```

	predicted_mpg	mpg	horsepower	weight	model year
0	15.447125	18	130	3504	70

	predicted_mpg	mpg	horsepower	weight	model year
1	14.053509	15	165	3693	70
2	15.785576	18	150	3436	70
...	...	...	...	...	...
389	32.4569	32	84	2295	82
390	30.354143	28	79	2625	82
391	29.726608	31	82	2720	82

```
392 rows × 5 columns
```

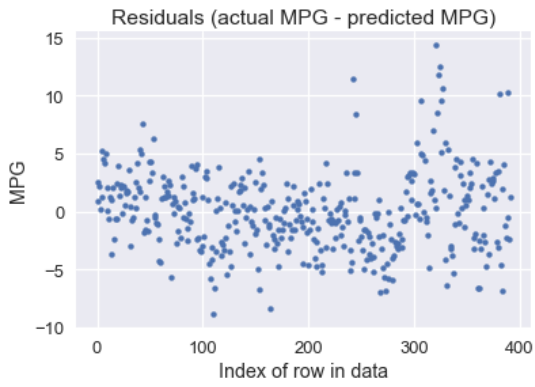
بما أننا أوجدنا $\boldsymbol{\theta}$ من النزول الاشتقاقي، يمكننا أن نتأكد من أول سطر في بياناتنا أن $\mathbf{X}_0 \cdot \boldsymbol{\theta}$ تطابق توقعنا السابق:

```
print(f'Prediction for first row: '
      f'{thetas[0] + thetas[1] * 130 + thetas[2] * 3504 + thetas[3] * 70:.2f}')
```

```
Prediction for first row: 15.45
```

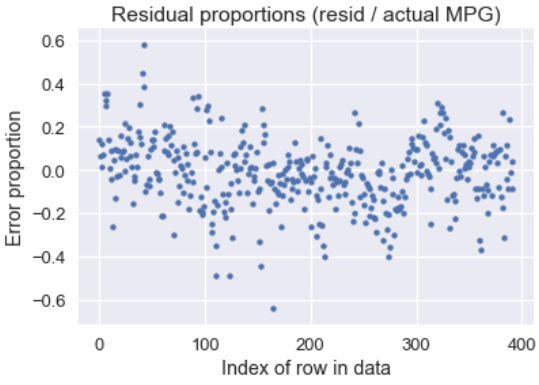
يمكننا رسم الفرق بين توقعنا والنتيجة الحقيقية (النتيجة الحقيقية - التوقع):

```
resid = y - linear_model(thetas, X)
plt.scatter(np.arange(len(resid)), resid, s=15)
plt.title('Residuals (actual MPG - predicted MPG)')
plt.xlabel('Index of row in data')
plt.ylabel('MPG');
```



يبدو واضحاً أن نموذجنا يعطي توقعات منطقية لكثير من السيارات، على الرغم أن بعض النتائج كان الفرق فيها أكثر من 10 ميل لكل جالون (بعض السيارات لديها أقل من 10!). قد يهمنا أكثر معرفة نسبة الخط بين التوقع والنتيجة الصحيحة لصرف الوقود:

```
resid_prop = resid / with_predictions['mpg']
plt.scatter(np.arange(len(resid_prop)), resid_prop, s=15)
plt.title('Residual proportions (resid / actual MPG)')
plt.xlabel('Index of row in data')
plt.ylabel('Error proportion');
```



يظهر لنا أن النموذج عادةً أبعد بحدود 20% من القيمة الصحيحة.

استخدام كامل البيانات

لاحظ أن في مثالنا حتى الآن، المصفوفة $\mathbf{X}$ تحتوي على أربع أعمدة: أولها يحتوي على القيمة 1 في جميع الصفوف، عمود قوة السيارة بالحصان، وزنها، وسنة الصناعة. ولكن، يسمح لنا النموذج باستخدام أكثر من هذا العدد:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

عندما نضيف المزيد من الأعمدة لمصفوفتنا، نقوم بتوسيع $\boldsymbol{\theta}$ لتحتوي على متغير لكل عمود في $\mathbf{x}$. بدلاً من اختيار فقط 3 أعمدة رقمية لإجراء التنبؤ، لماذا لا نستخدم جميع الأعمدة السبعة؟

```
cols = ['bias', 'cylinders', 'displacement', 'horsepower',
        'weight', 'acceleration', 'model year', 'origin']
X = mpg_mat[cols].to_numpy()
mpg_mat[cols]
```

	bias	cylinders	displacement	horsepower	weight	acceleration	model year	origin
0	1	8	307	130	3504	12	70	1
1	1	8	350	165	3693	11.5	70	1
2	1	8	318	150	3436	11	70	1
...	...	...	...	...	...	...	...	...
389	1	4	135	84	2295	11.6	82	1
390	1	4	120	79	2625	18.6	82	1
391	1	4	119	82	2720	19.4	82	1

392 rows × 8 columns

```
thetas_all = minimize(mse_loss, grad_mse_loss, X, y, progress=10)
print(f'theta: {thetas_all} | loss: {mse_loss(thetas_all, X, y):.2f}')
```

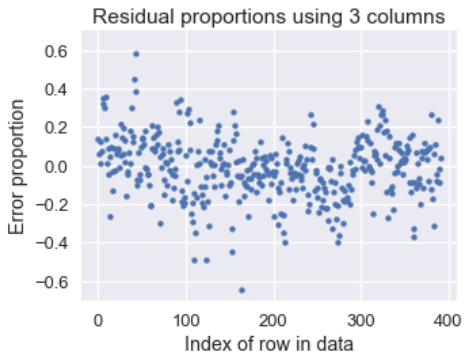
```
theta: [0. 0. 0. 0. 0. 0. 0. 0.] | loss: 610.47
theta: [-0.5  -0.81  0.02 -0.04 -0.01 -0.07  0.59  1.3 ] | loss: 11.22
theta: [-17.23  -0.49   0.02  -0.02  -0.01   0.08   0.75   1.43] | loss: 10.85
theta: [-17.22  -0.49   0.02  -0.02  -0.01   0.08   0.75   1.43] | loss: 10.85
```

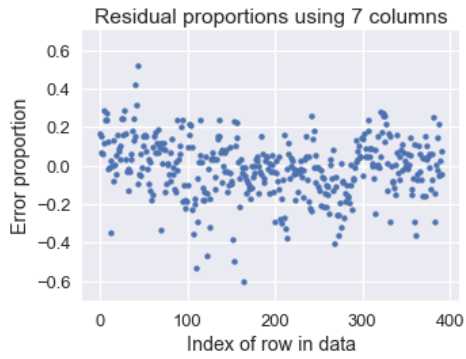
وفقاً لنتيجة النزول الاشتقاقي، فإن النموذج الخطي يمكننا تعريفه كالتالي:

$$y = -17.22 - 0.49x_1 + 0.02x_2 - 0.02x_3 - 0.01x_4 + 0.08x_5 + 0.75x_6 + 1.43x_7$$

نلاحظ أن خسارتنا قلت من 11.6 باستخدام ثلاث أعمدة إلى 10.85 باستخدام جميع الأعمدة الرقمية السبعة في بياناتنا. سنرى نسبة الخطأ في الرسم البياني لكل التوقعين السابق (باستخدام ثلاث أعمدة) والجديد (باستخدام سبع أعمدة):

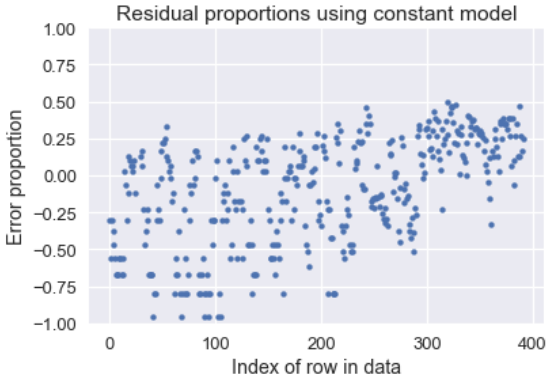
```
resid_prop_all = (y - linear_model(thetas_all, X)) / with_predictions['mpg']
plt.figure(figsize=(10, 4))
plt.subplot(121)
plt.scatter(np.arange(len(resid_prop)), resid_prop, s=15)
plt.title('Residual proportions using 3 columns')
plt.xlabel('Index of row in data')
plt.ylabel('Error proportion')
plt.ylim(-0.7, 0.7)

plt.subplot(122)
plt.scatter(np.arange(len(resid_prop_all)), resid_prop_all, s=15)
plt.title('Residual proportions using 7 columns')
plt.xlabel('Index of row in data')
plt.ylabel('Error proportion')
plt.ylim(-0.7, 0.7)

plt.tight_layout();
```





على الرغم أن الفرق بسيط، نلاحظ أن الفرق أقل عندما نستخدم السبع أعمدة. كلا النموذجين أفضل من النموذج الثابت، كما يوضح الرسم البياني التالي:

```
constant_resid_prop = (y - with_predictions['mpg'].mean()) / with_predictions['mpg']
plt.scatter(np.arange(len(constant_resid_prop)), constant_resid_prop, s=15)
plt.title('Residual proportions using constant model')
plt.xlabel('Index of row in data')
plt.ylabel('Error proportion')
plt.ylim(-1, 1);
```



استخدام النموذج الثابت وصلت فيه نتائج الخطأ إلى أكثر من 75% لكثير من السيارات!

ملخص الانحدار الخطي المتعدد

تعرفنا على النموذج الخطي للانحدار. على عكس النموذج الثابت، الانحدار الخطي يأخذ خصائص من البيانات بالحسبان عند إجراء التوقعات، مما يجعله أكثر فائدة عندما يكون لدينا علاقات في بياناتنا.

خطوات ضبط النموذج من المفترض أن تكون واضحة الآن:

- اختيار النموذج.
- اختيار دالة الخسارة.
- تقليل دالة الخسارة باستخدام النزول الاشتقاقي.

من المفيد معرفة أن بإمكاننا التعديل على أحد المكونات دون الأخرى. في هذا الجزء، تعرفنا على النموذج الخطي دون التغير في دالة الخسارة أو استخدام خوارزميات تقليل أخرى. على الرغم أن النمذجة قد تكون معقدة، يكون أسهل التركيز على مكون واحد فقط في كل مرة، ثم جمع المكونات مع بعضها البعض.

المربعات الصغرى - منظور هندسي

لنتذكر أننا أوجدنا المتغيرات الرياضية المثالية للنموذج الخطي بواسطة التحسين من دالة الخسارة باستخدام النزول الاشتقاقي. ذكرنا أيضاً أن الانحدار الخطي للمربعات الصغرى يمكن حسابه تحليلياً. على الرغم أن النزول الاشتقاقي أكثر عملياً، هذا المنظور الهندسي سيساعدك على فهم الانحدار الخطي بشكل أكبر.

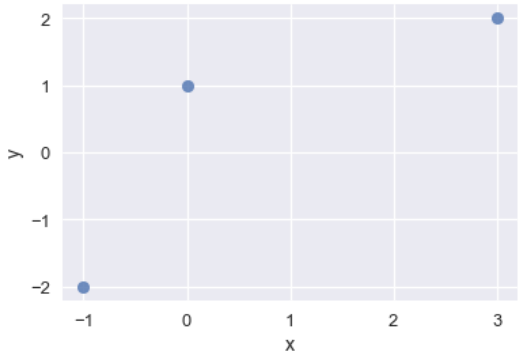
يتوقع من القارئ أن يكون على علم بفضاء المتجهات Vector Space وطريقة القيام بالعمليات الحسابية عليها.

لنفترض أننا نبحث عن النموذج الخطي للبيانات التالية:

x	y
3	2

x	y
0	1
1-	2-

```
data = pd.DataFrame(
    [
        [3,2],
        [0,1],
        [-1,-2]
    ],
    columns=['x', 'y']
)

sns.regplot(x='x', y='y', data=data, ci=None, fit_reg=False);
```



لنفترض أن النموذج المثالي هو النموذج بأقل خسارة، وان خطأ المربعات الصغرى هي أداة مقبولة للقياس.

المربعات الصغرى: النموذج الثابت

كما فعلنا في بيانات الإكراميات، لنبدأ بالنموذج الثابت: نموذج يتوقع رقم واحد فقط.

$$\theta = C$$

نعمل نحن فقط مع قيم y:

y
2
1
2-

وهدفنا هو إيجاد قيمة θ التي تنتج لنا خطأ يقلل من الخسارة التربيعية:

$$L(\theta, \mathbf{y}) = \sum_{i=1}^{n} (y_i - \theta)^2$$

لنتذكر أن للنموذج الثابت، قيمة θ التي تقلل الخسارة في دالة الخطأ التربيعي المتوسط MSE هي $\bar{\mathbf{y}}$، متوسط قيم $\mathbf{y}$. يمكن إيجاد العملية الحسابية الكاملة في جزء دوال الخسارة في فصل النماذج والتوقعات.

لاحظ أن دالة الخسارة لدينا هي مجمع التربيع. القاعدة L2 للمتجهات هي أيضاً مجموع التربيع، ولكن مع الجذر التربيعي:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

إذا جعلنا $y_i - \theta = v_i$:

$$\begin{aligned} L(\theta, \mathbf{y}) &= v_1^2 + v_2^2 + \cdots + v_n^2 \\ &= \|\mathbf{v}\|^2 \end{aligned}$$

يعني ذلك أن بإمكاننا تعريف الخسارة على أنها تربيع القاعدة L2 لمتّجه $\mathbf{v}$. يمكننا وصف v_i كالتالي $\forall i \in [1, n]$ إذاً ذلك في كتعريف $y_i - \theta$ ديكارتي:

$$
\begin{aligned}
\mathbf{v} &= \begin{bmatrix} y_1 - \theta \\ y_2 - \theta \\ \vdots \\ y_n - \theta \end{bmatrix} \\
&= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \theta \\ \theta \\ \vdots \\ \theta \end{bmatrix} \\
&= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \theta \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}
\end{aligned}
$$

إذاً، يمكننا كتابة دالة الخسارة كالتالي:

$$
\begin{aligned}
L(\theta, \mathbf{y}) &= \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \theta \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\|^2 \\
&= \| \mathbf{y} - \hat{\mathbf{y}} \|^2
\end{aligned}
$$

الوصف $\theta \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ هو مضاعف عددي للأعمدة في المتّجه $\mathbf{1}$، وهي أيضاً نتيجة التوقع، ويرمز لها $\hat{\mathbf{y}}$.

يعطينا ذلك منظوراً جديداً عن معنى تقليل الخسارة في خطأ المربعات الصغرى.

قيم $\mathbf{y}$ و $\mathbf{1}$ ثابتة، ولكن θ يمكن أن تأخذ أي قيمة، لذا $\hat{\mathbf{y}}$ يمكن أن تكون أي مضاعف عددي ل $\mathbf{1}$. نريد إيجاد θ لتكون $\theta\mathbf{1}$ أقرب ما تكون إلى $\mathbf{y}$. نستخدم $\hat{\theta}$ لوصف ذلك الضبط المثالي ل θ.

المربعات الصغرى: نموذج خطي بسيط

الآن، لنلقي نظره على نموذج الانحدار الخطي البسيط. يشبه النموذج بشكل كبير لاشتقاق النموذج الثابت، ولكن لاحظ الفرق وفكر بطريقة عامة لإجراء الانحدار الخطي المتعدد.

النموذج الخطي البسيط هو:

$$f_{\boldsymbol{\theta}}(x_i) = \theta_0 + \theta_1 x_i$$

هدفنا إيجاد $\boldsymbol{\theta}$ التي تنتج لنا خطأً بأقل خطأ تربيعي:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) &= \sum_{i=1}^{n} (y_i - f_{\boldsymbol{\theta}}(x_i))^2 \\
&= \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 x_i)^2 \\
&= \sum_{i=1}^{n} (y_i - \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix})^2
\end{aligned}
$$

لمساعدتنا على تحويل شكل جمع الخسارة إلى شكل مصفوفة، دعنا نوسع من الخسارة ب $n = 3$:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = &(y_1 - \begin{bmatrix} 1 & x_1 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix})^2 \\
&+ (y_2 - \begin{bmatrix} 1 & x_2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix})^2 \\
&+ (y_3 - \begin{bmatrix} 1 & x_3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix})^2
\end{aligned}
$$

مرة أخرى، دالة الخسارة هي مجموع التربيع و القاعدة L2 للمتّجه هي الجذر التربيعي لمجموع التربيع:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

إذا جعلنا $v_i = y_i - \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$:

$$
\begin{aligned}
L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) &= v_1^2 + v_2^2 + \cdots + v_n^2 \\
&= \|\mathbf{v}\|^2
\end{aligned}
$$

كما في السابق، يمكننا وصف خسارتنا على أنها تربيع القاعدة L2 للمتّجه $\mathbf{v}$.

$$
v_i = y_i - \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \forall i \in [1, 3]
$$

$$
\begin{aligned}
L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) &= \left\| \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad - \quad \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \right\|^2 \\
&= \left\| \quad \mathbf{y} \quad - \quad \mathbf{X} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad \right\|^2 \\
&= \| \quad \mathbf{y} \quad - \quad f_{\boldsymbol{\theta}}(\mathbf{x}) \quad \|^2 \\
&= \| \quad \mathbf{y} \quad - \quad \hat{\mathbf{y}} \quad \|^2
\end{aligned}
$$

عملية ضرب المصفوفة $\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ هي تركيبة خطية للأعمدة في $\mathbf{X}$: كل θ_i يتم ضربها بعمود واحد من $\mathbf{X}$، يظهر لنا هذا المنظور أن $f_{\boldsymbol{\theta}}$ تركيبة خطية للخصائص في بياناتنا.

$\mathbf{X}$ و $\mathbf{y}$ ثابتين، ولكن θ_0 و θ_1 يمكن أن يأخذان أي قيمه، لذا $\hat{\mathbf{y}}$ يمكن أن تأخذ أياً من التركيبات الخطية من الأعمدة في $\mathbf{X}$. للحصول على أقل خسارة، تريد أن نختار $\boldsymbol{\theta}$ التي فيها $\hat{\mathbf{y}}$ أقرب ما تكون إلى $\mathbf{y}$، يرمز لها بالرمز $\hat{\boldsymbol{\theta}}$.

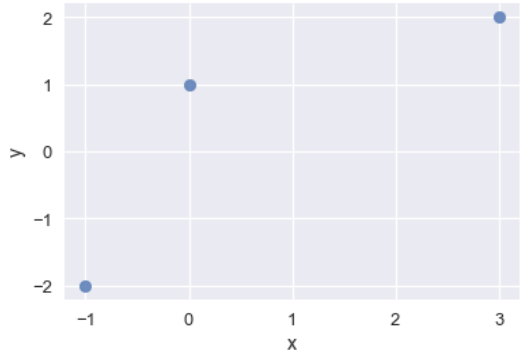
الفكرة الهندسية

الآن، لنحاول تكوين فكرة عن أهمية أن تكون $\hat{\mathbf{y}}$ محدودة للتركيبات الخطية للأعمدة في $\mathbf{X}$. على الرغم أن مدى أي متّجه يحتوي على عدد غير محدود من التركيبات الخطية، غير محدود لا تعني أياً كان، التركيبات الخطية محدودة بواسطة أساس المتّجه.

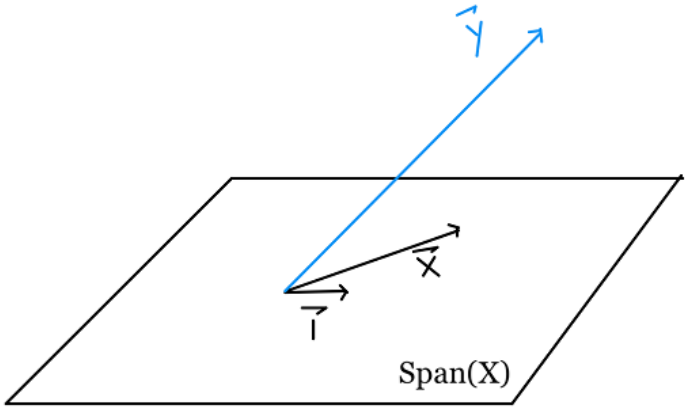
للتذكير، هذه دالة الخسارة ومخطط التشتت:

$$
L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) \quad = \quad \| \quad \mathbf{y} \quad - \quad \mathbf{X}\boldsymbol{\theta} \quad \|^2
$$

```
sns.regplot(x='x', y='y', data=data, ci=None, fit_reg=False);
```



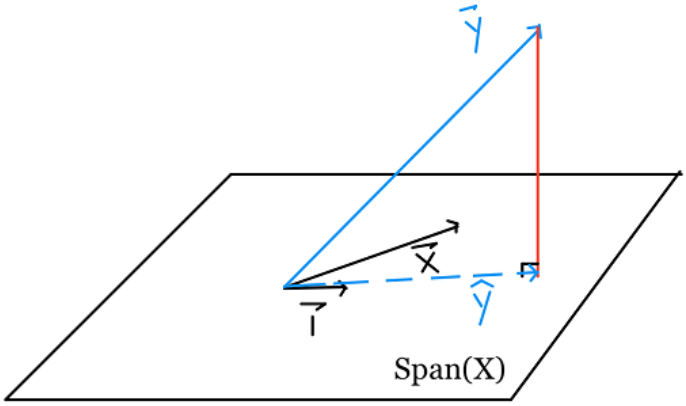
من خلال مشاهدتنا لمخطط التشتت، نلاحظ أنه لا يوجد خط مثالي للنقاط، لذا لن نستطيع الوصول إلى خسارة تساوي صفر. نعرف أن $\mathbf{y}$ ليست على مستوى خطي مع $\mathbf{x}$ و $ \textbf{1}\ $:



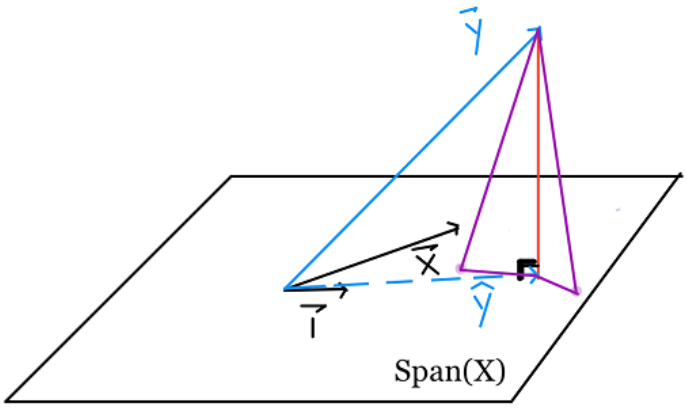
بما أن حساب الخسارة يكون بالمسافة، يمكننا ملاحظة أنه لتقليل الخسارة $L(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2$، فنريد أن تكون $\mathbf{X}\boldsymbol{\theta}$ أقرب ما تكون إلى $\mathbf{y}$.

رياضياً، نرى توقع $\mathbf{y}$ في فضاء المتّجه الممتد بواسطة الأعمدة في $\mathbf{X}$، لأن التوقع لأي متّجه هي أقرب نقطة في $Span(\mathbf{X})$ للمتّجه. لذا، اختيار $\boldsymbol{\theta}$ يكون فيها $proj_{Span(\mathbf{X})}\mathbf{y} = \mathbf{X}\boldsymbol{\theta} = \hat{\mathbf{y}}$ هو الحل المثالي.

> التوقع = proj = projection



لنعرف لماذا، لنأخذ بالاعتبار النقاط الأخرى في فضاء المتّجه، النقاط باللون البنفسجي:

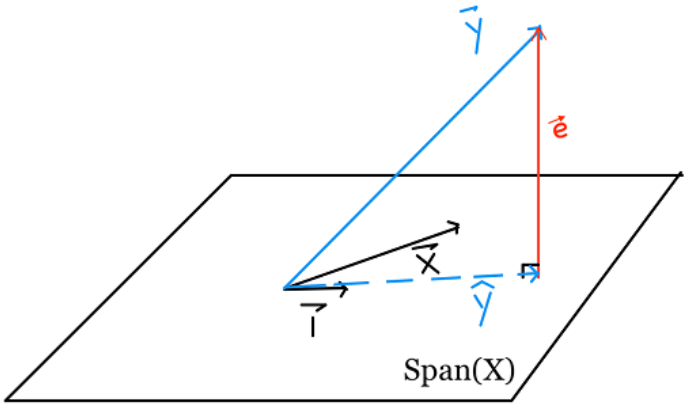


بناءًا على نظرية فيثاغورس، أي نقطة على السطح هي أبعد عن $\mathbf{y}$ من $\hat{\mathbf{y}}$. طول العمود المقابل ل $\hat{\mathbf{y}}$ هو خطأ المربعات الصغرى.

الجبر الخطي

تحدثنا بشكل كبير عن الجبر الخطي، ما تبقى لنا الآن هو حل $\hat{\boldsymbol{\theta}}$ التي تكون لنا ما نبحث عنه $\hat{\mathbf{y}}$

بعض النقاط نأخذها بالاعتبار:



- $\hat{\mathbf{y}} + \mathbf{e} = \mathbf{y}$
- $\mathbf{e}$ متوازيه عامودياً مع $\mathbf{x}$ و $\mathbf{1}$
- $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\theta}}$ هي المتّجه الأقرب إلى $\mathbf{y}$ في فضاء المتّجهات الممتدة من $\mathbf{x}$ و $\mathbf{1}$

ولذا، تنتج لنا المعادلة التالية:

$$\mathbf{X}\hat{\boldsymbol{\theta}} + \mathbf{e} = \mathbf{y}$$

ضرب الجهة اليسرى لكل القيم ب $\mathbf{X}^T$ ينتج لنا التالي:

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\theta}} + \mathbf{X}^T\mathbf{e} = \mathbf{X}^T\mathbf{y}$$

بما أن $\mathbf{e}$ متعامدة مع الأعمدة في $\mathbf{X}$، فإن $\mathbf{X}^T\mathbf{e}$ هو عمود متّجه يحتوي على أصفار. لذا، نصل إلى المعادلة التالية:

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}^T\mathbf{y}$$

من هنا، يمكننا بسهولة الحل لإيجاد $\hat{\boldsymbol{\theta}}$ بواسطة ضرب الجهة اليسرى في كلا الجانبين ب $(\mathbf{X}^T\mathbf{X})^{-1}$:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

ملاحظة: يمكننا الحصول على نفس النتيجة بواسطة التقليل باستخدام متجهات التفاضل والتكامل، ولكن بالنسبة لخطأ المربعات الصغرى، متجهات التفاضل والتكامل ليست ضرورية. لدوال الخسارة الأخرى، سنحتاج لاستخدام متجهات التفاضل والتكامل للحصول على النتيجة التحليلية.

إنهاء الدراسة

لنعود لتجربتنا، لنطبق ما تعلمناه، ونشرح إجابتنا:

$$\mathbf{y} = \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}$$

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \left(\begin{bmatrix} 1 & 1 & 1 \\ 3 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 1 & 0 \\ 1 & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 3 & 0 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix} \\
&= \left(\begin{bmatrix} 3 & 2 \\ 2 & 10 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 8 \end{bmatrix} \\
&= \frac{1}{30-4} \begin{bmatrix} 10 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 8 \end{bmatrix} \\
&= \frac{1}{26} \begin{bmatrix} -6 \\ 22 \end{bmatrix} \\
&= \begin{bmatrix} -\frac{3}{13} \\ \frac{11}{13} \end{bmatrix}
\end{aligned}$$

قمنا تحليلياً بإيجاد النموذج المثالي لانحدار المربعات الصغرى وهو $f_{\theta}(x_i) = -\frac{3}{13} + \frac{11}{13}x_i$. نعرف أن اختيارنا ل $\boldsymbol{\theta}$ صحيح باستخدام الخاصية الرياضية التي تقول توقع $\mathbf{y}$ لامتداد الأعمدة في $\mathbf{X}$ ينتج لنا أقرب نقطة في فضاء المتّجه ل $\mathbf{y}$. تحت القيود الخطية باستخدام خسارة المربعات الصغرى، الحل ل $\hat{\boldsymbol{\theta}}$ ينتج لنا توقع مضمون أنه الحل الأفضل.

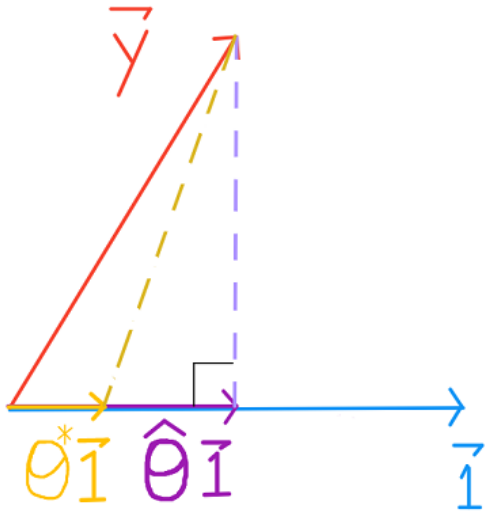
عندما تعتمد المتغيرات خطياً

لكل متغير إضافي، نضيف عمود جديد إلى $\mathbf{X}$. امتداد أعمدة ل $\mathbf{X}$ هو التركيب الخطي لأعمدة المتّجهات، لذا إضافة أعمدة تغير من الامتداد فقط إذا كانت مستقلة خطياً عن بقية الأعمدة الموجودة مسبقاً.

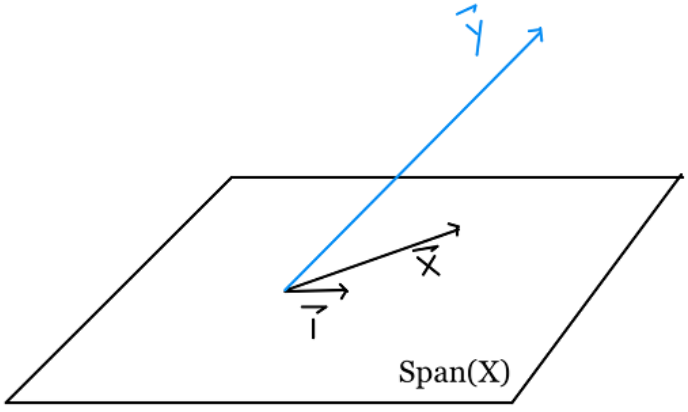
عندما يكون العمود المضاف غير مستقل خطياً، يمكن وصفة كتركيب خطي من أحد الأعمدة الأخرى، لذا، لن يكون لنا أي متّجه جديده في الفضاء الجزئي.

لنتذكر أن امتداد $\mathbf{X}$ مهم لأنه الفضاء الجزئي الذي نريد أن نتوقع $\mathbf{y}$ فيه. إذا لم يتغير هذا الفضاء، فإن التوقع لن يتغير.

مثلاً، عندما عرفنا $\mathbf{x}$ على النموذج الثابت لنحصل على النموذج الخطي البسيط، عرفنا دالة مستقلة. $\mathbf{x} = \begin{bmatrix} 3 \\ 0 \\ -1 \end{bmatrix}$ لا يمكن وصفها كعدديه من $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. لذا، انتقلنا من إيجاد توقع ل $\mathbf{y}$ في الخط:



إلى إيجاد توقع ل $\mathbf{y}$ على السطح:

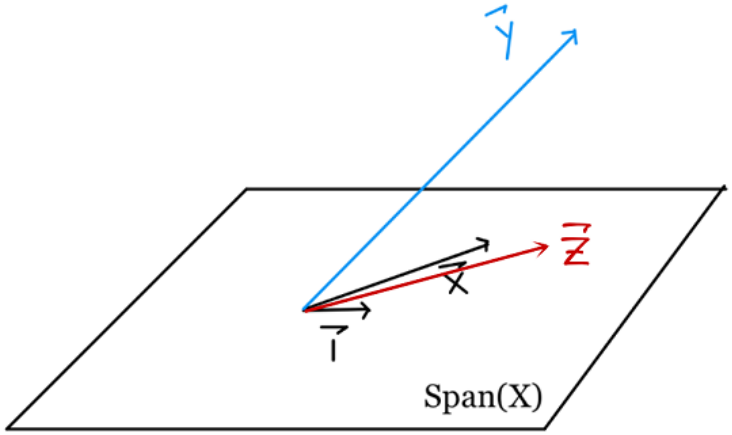


الآن، لنعرف متغير آخر، $\mathbf{z}$، يكون بشكل واضح هذا العمود متحيز:

z	1	x	y
4	1	3	2
1	1	0	1

z	1	x	y
0	1	1-	2-

لاحظ أن $\mathbf{z} = \mathbf{1} + \mathbf{x}$. بما أن $\mathbf{z}$ تركيبتها خطية من $\mathbf{1}$ و $\mathbf{x}$، فأنها تقع في $Span(\mathbf{X})$. الآن، $\mathbf{z}$ معتمده خطياً على $\{\mathbf{1}, \mathbf{x}\}$ ولا تغيّر في $Span(\mathbf{X})$. لذا، توقع $\mathbf{y}$ في الفضاء الجزئي الممتد بواسطة $\mathbf{1}$، $\mathbf{x}$ و $\mathbf{z}$ سيكون مطابق لتوقع $\mathbf{y}$ في الفضاء الجزئي الممتد بواسطة $\mathbf{1}$ و $\mathbf{x}$.



يمكننا ايضاً ملاحظة ذلك عند تقليل دالة الخسارة:

$$L(\boldsymbol{\theta}, \mathbf{d}, \mathbf{y}) = \left\| \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - \begin{bmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \right\|^2$$

الحل المتوقع لنا سيكون كالشكل التالي $\theta_0\mathbf{1} + \theta_1\mathbf{x} + \theta_2\mathbf{z}$.

بما أن $\mathbf{z} = \mathbf{1} + \mathbf{x}$، ايضاً كانت θ_0، θ_1 و θ_2، القيم المتوقعة يمكن إعادة كتابتها كالتالي:

$$\theta_0\mathbf{1} + \theta_1\mathbf{x} + \theta_2(\mathbf{1} + \mathbf{x}) = (\theta_0 + \theta_2)\mathbf{1} + (\theta_1 + \theta_2)\mathbf{x}$$

إذاً، إضافة $\mathbf{z}$ لن تغير أي شيء. الفرق الوحيد هو أن بإمكاننا وصف هذا التوقع بعدة أشكال. لنتذكر أننا أوجدنا توقع $\mathbf{y}$ على امتداد $\mathbf{1}$ و $\mathbf{x}$ وكان:

$$\begin{bmatrix} \mathbf{1} & \mathbf{x} \end{bmatrix} \begin{bmatrix} -\frac{3}{13} \\ \frac{11}{13} \end{bmatrix} = -\frac{3}{13}\mathbf{1} + \frac{11}{13}\mathbf{x}$$

لكن، مع التعرف على $\mathbf{z}$، يمكننا وصف توقع المتّجه بأكثر من طريقه.

بما أن $\mathbf{1} = \mathbf{z} - \mathbf{x}$، ف $\hat{\mathbf{y}}$ يمكن وصفها كالتالي:

$$-\frac{3}{13}(\mathbf{z} - \mathbf{x}) + \frac{11}{13}\mathbf{x} = -\frac{3}{13}\mathbf{z} + \frac{14}{13}\mathbf{x}$$

بما أن $\mathbf{x} = \mathbf{z} + \mathbf{1}$، ف $\hat{\mathbf{y}}$ يمكن وصفها كالتالي:

$$-\frac{3}{13}\mathbf{1} + \frac{11}{13}(\mathbf{z} + \mathbf{1}) = \frac{8}{13}\mathbf{1} + \frac{11}{13}\mathbf{z}$$

ولكن جميع الأوصاف الثلاثة تقدم نفس التوقع.

في الختام، إضافة عمود غير مستقل خطياً إلى $\mathbf{x}$ لا يغير في $Span(\mathbf{X})$، ولذلك لن يغير في التوقع والحل لمشكلة المربعات الصغرى.

طريقتين للتفكير

أضفنا مخططات التشتت مرتين في هذا الدرس. أول مرة للتذكير أنه كما في السابق، نحاول إيجاد الخط المثالي للبيانات. المرة الثانية كانت تأكد أنه لا يوجد خط مثالي لجميع النقاط. بصرف النظر هن ذلك، حاولنا عدم تخريب رسم فضاء المتّجه بمخططات التشتت. ذلك لأن مخططات التشتت تتوافق مع نظرية مساحة الصف في مشكلة المربعات الصغرى: النظر لكل نقاط البيانات ومحاولة التقليل في المسافة بين توقعنا و كل نقطة. في هذا الدرس، تعرفنا على نطريقة مساحة العامود: كل متغير كان متّجه، تكون لنا فضاء من الإجابات المُحتملة (التوقعات).

تطبيق عملي للانحدار الخطي

في هذا الجزء، سنقوم بتطبيق عملي لنموذج الانحدار الخطي على بيانات. لدى البيانات التي سنعمل عليها العديد من الخصائص، مثل الطول والحجم لحيوان الحِمار.

مهمتنا هي توقع الوزن باستخدام الانحدار الخطي.

نظره عامه على البيانات

سنبدأ أولاً بقراءة البيانات وأخذ نظره سريعة على محتواها:

لتحميل البيانات، اضغط هنا.

```
import pandas as pd

donkeys = pd.read_csv("donkeys.csv")
donkeys.head()
```

	BCS	Age	Sex	...	Height	Weight	WeightAlt
0	3	<2	stallion	...	90	77	NaN
1	2.5	<2	stallion	...	94	100	NaN
2	1.5	<2	stallion	...	95	74	NaN
3	3	<2	female	...	96	116	NaN
4	2.5	<2	female	...	91	91	NaN

```
5 rows × 8 columns
```

فكرة جيدة دائماً هي النظر *لعدد* البيانات لدينا عن طريق عرض مقاسات الـ DataFrame. عندما يكون حجم البيانات لدينا كبير، طباعة كامل البيانات قد يسبب مشاكل للمتصفح وتتسبب بإغلاقه:

```
donkeys.shape
```

```
(544, 8)
```

البيانات قليله نسبياً، لدينا فقط 544 سطر و 8 أعمدة. لنرى ما هي الأعمدة المتوفرة لدينا:

```
donkeys.columns.values
```

```
array(['BCS', 'Age', 'Sex', 'Length', 'Girth', 'Height', 'Weight',
       'WeightAlt'], dtype=object)
```

فهم البيانات لدينا يساعدنا على توجيه عملية تحليلنا لها، يجب علينا فهم ما يحتويه كل عمود. بعض هذه الأعمدة واضحة من مسمياتها، ولكن بعضها يحتاج شرحاً أكثر:

- `BCS` (Body Condition Score): حالة الجسم (تقييم لصحة جسد الحيوان).
- `Grith`: مقياس لمتوسط جسم الحيوان.
- `WeightAlt`: وزن آخر (من بين البيانات لدينا تم وزن بعضهم مرتين وعددهم 31، تم وزنهم مرتين للتأكد من دقة الوزن).

فكرة مناسبة الآن هو تحديد أي الأعمدة تحتوي على بيانات كمية وأيها يحتوي على بيانات أسميه.

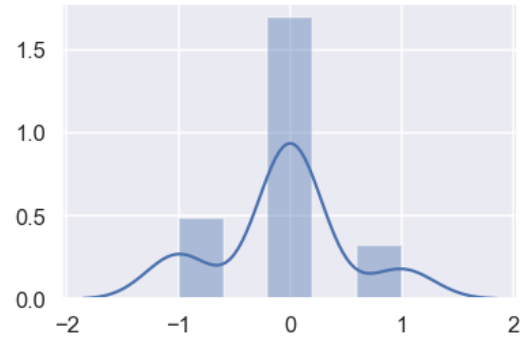
الكمية: `Weight`، `Height`، `Girth`، `Length` و `WeightAlt`.
الاسمية: `Age`، `BCS` و `Sex`.

تنظيف البيانات

في هذا الجزء، سنتحقق من البيانات إذا كانت تحتوي على أي شذوذ ونتعامل معها.

عن طريق التحقق من عمود `WeightAlt`، يمكننا التأكد من دقة الوزن بأخذ الفرق بين الوزنين ورسمها النتائج:

```
difference = donkeys['WeightAlt'] - donkeys['Weight']
sns.distplot(difference.dropna());
```



الأوزان تبدو معقولة بفارق 1 أو أقل كيلو قرام فيما بينها.

الآن، يمكننا النظر إلى قيم غريبه والتي قد تظر أن لدينا مشاكل أو أخطاء. يمكننا استخدام الدالة Quantile للكشف عن القيم الشاذه:

```
donkeys.quantile([0.005, 0.995])
```

	BCS	Length	Girth	Height	Weight	WeightAlt
0.005	1.5	71.145	90	89	71.715	98.75
0.995	4	111	131.285	112	214	192.8

شرح مبسط للدالة quantile:
تقوم الدالة بإظهار نتائج بناءًا على النسبة المعطاة لها، في المثال السابق طلبنا نتائج من هم في النسبة 0.005 وأقل و منهم في النسبة 0.995 وأقل.
مثلاً في العمود Height، يظهر لنا أن 0.5% أو أقل يبلغ طولهم 89.0 بينما 99.5% أو أقل يبلغ طولهم 112.0 في بياناتنا.

لكل الأعمدة الكمية، يمكن أن نبحث عن أي سطر تكون نتيجته خارج هذه الأعداد، لأننا نريد أن يطبق النموذج على أي حيوان صحته ممتازة وناضج.

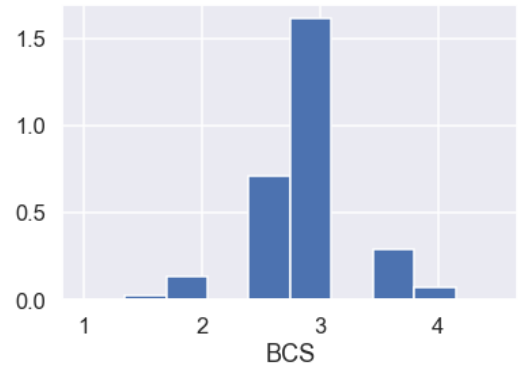
أولاً، لنرى العمود BCS:

```
donkeys[(donkeys['BCS'] < 1.5) | (donkeys['BCS'] > 4)]['BCS']
```

```
291    4.5
445    1.0
Name: BCS, dtype: float64
```

أيضاً عند النظر على المخطط الشريطي للعمود BCS:

```
plt.hist(donkeys['BCS'], density=True)
plt.xlabel('BCS');
```



بالأخذ بالإعتبار أن العمود BCS يظهر لنا مدى صحة ونضوج الحيوان، حصول الحيوان على 1 يبين أنه هزيل وحصوله على 4.5 يبين أن وزنه زائد.
أيضاً بالنظر إلى المخطط الشريطي، فقط يظهر لدينا اثنين بقيم شاذه في العمود BCS. لذا سنقوم بحذف هذه البيانات.

الآن، لنلقي نظره على الأعمدة Girth، Length و Height:

```
donkeys[(donkeys['Length'] < 71.145) | (donkeys['Length'] > 111)]['Length']
```

```
8       46
22      68
26      69
216    112
Name: Length, dtype: int64
```

```
donkeys[(donkeys['Girth'] < 90) | (donkeys['Girth'] > 131.285)]['Girth']
```

```
8       66
239    132
283    134
523    134
Name: Girth, dtype: int64
```

```
donkeys[(donkeys['Height'] < 89) | (donkeys['Height'] > 112)]['Height']
```

```
8       71
22      86
244    113
523    116
Name: Height, dtype: int64
```

بالنسبة لهذه الأعمدة الثلاثة، يبدو أن السطر 8 يحتوي على قيم ادنى من باقي البيانات بينما الأسطر الأخرى قريبه جداً من باقي البيانات ولا يحتاج أن نحذفها.

أخيراً، لنرى العمود Weight:

```
donkeys[(donkeys['Weight'] < 71.715) | (donkeys['Weight'] > 214)]['Weight']
```

```
8       27
26      65
50      71
291    227
523    230
Name: Weight, dtype: int64
```

أو سطرين وآخر سطرين بعيدة جداً عن بقية البيانات وعلى الأرجح سنقوم بحذفها. يمكننا أن نبقي السطر في المنتصف كونه قريباً جداً من بقية البيانات.

بما أن العمود WeightAlt مشابه للعمود Weight، لن نقوم بالبحث عن شواذ فيه. ولتجميع ما فهمناه، هذا ما سنقوم بفلترته في بياناتنا:

- الإبقاء على البيانات بين 1.5 و 4 في العمود BCS.
- الإبقاء على الأسطر التي بياناتها بين 71 و 214 في العمود Weight.

```
donkeys_c = donkeys[(donkeys['BCS'] >= 1.5) & (donkeys['BCS'] <= 4) &
                    (donkeys['Weight'] >= 71) & (donkeys['Weight'] <= 214)]
```

فصل بيانات التدريب والاختبار

قبل أن نبدأ بتحليل البيانات، نريد أن نقسم البيانات إلى تقسيم 80/20، نستخدم فيها 80% من البيانات لتدريب النموذج، ونترك 20% لتقييم واختبار النموذج:

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(donkeys_c.drop(['Weight'], axis=1),
                                                    donkeys_c['Weight'],
                                                    test_size=0.2,
                                                    random_state=42)
X_train.shape, X_test.shape
```

```
((431, 7), (108, 7))
```

أستخدم الكاتب دالة train_test_split والتي تأتي من مكتبة sklearn وهي مكتبة متخصصه بأدوات تحليل البيانات وتعلم الآله.

نقوم أيضاً بإنشاء دالة تقييم التوقع على بيانات الإختبار، لنستخدم الخطأ التربيعي المتوسط:

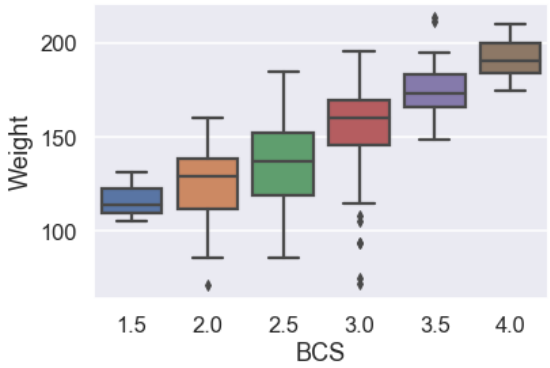
```
def mse_test_set(predictions):
    return float(np.sum((predictions - y_test) ** 2))
```

استكشاف البيانات وتصويرها

كالمعتاد، سنتحقق من البيانات قبل محاولة ضبط النموذج عليها.

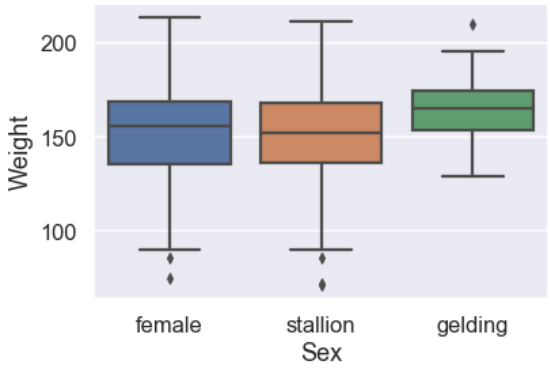
أولاً، لنطلع على البيانات الاسمية باستخدام مخطط الصندوق:

```
sns.boxplot(x=X_train['BCS'], y=y_train);
```



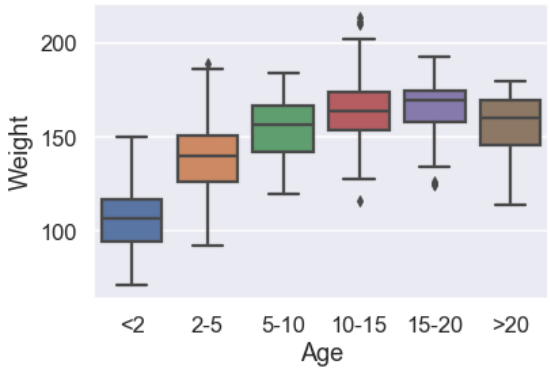
يبدو أن متوسط الوزن يزداد مع BCS، وليس خطياً:

```
sns.boxplot(x=X_train['Sex'], y=y_train,
            order = ['female', 'stallion', 'gelding']);
```



يظهر أن الجنس لا يتأثر كثيراً بالوزن

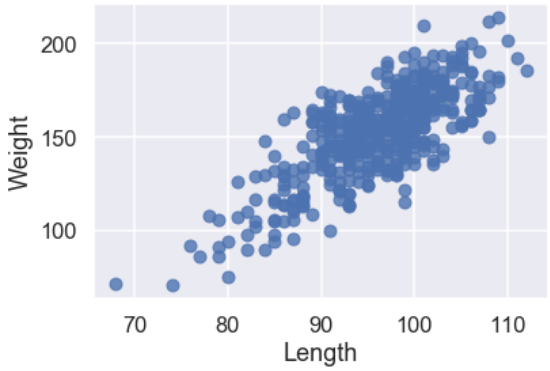
```
sns.boxplot(x=X_train['Age'], y=y_train,
            order = ['<2', '2-5', '5-10', '10-15', '15-20', '>20']);
```



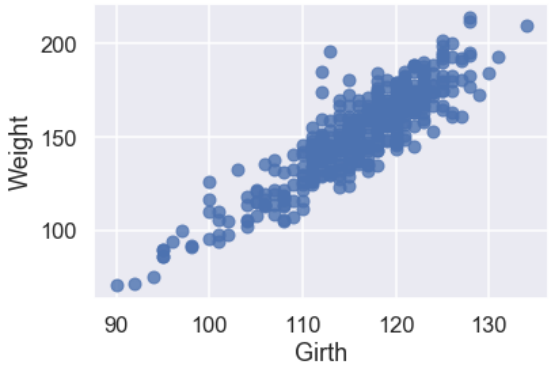
بعمر 5 أو أكثر، توزيع الأوزان لا يظهر أي تغير كبير.

الآن، لنلقي نظره على البيانات الكمية. يمكننا رسم كل واحدٍ منها مع المتغير الذي نريد التنبؤ عنه:

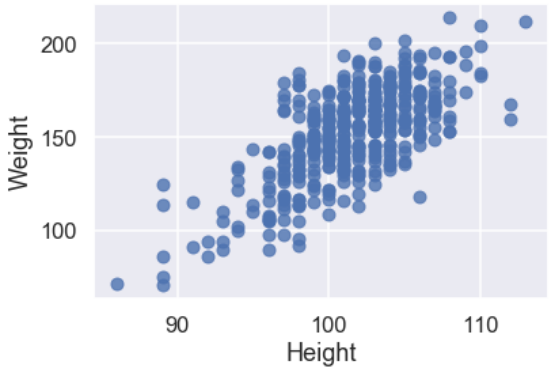
```
X_train['Weight'] = y_train
sns.regplot('Length', 'Weight', X_train, fit_reg=False);
```




```
sns.regplot('Girth', 'Weight', X_train, fit_reg=False);
```



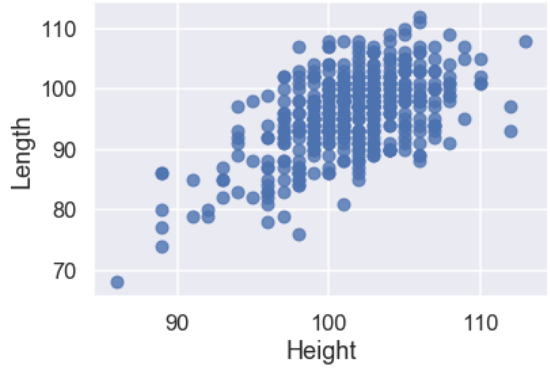

```
sns.regplot('Height', 'Weight', X_train, fit_reg=False);
```



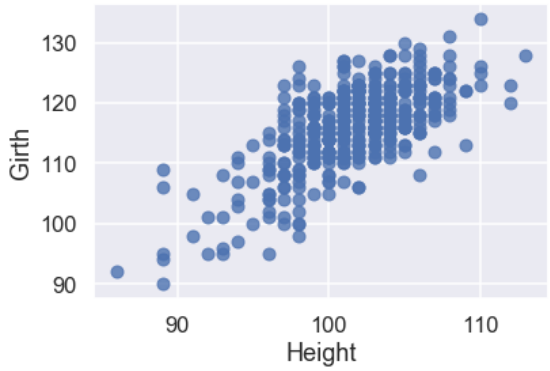
جميع البيانات الكمية لديها علاقة خطية مع المتغير الذي نريد توقعه `Weight`، لذا لا نحتاج للقيام بأي تعديلات على البيانات التي سندرب النموذج عليها.

أيضاً تبدو فكرة جيدة إذا رأينا بين كل متغير وآخر إذا كانت العلاقة بينهم خطية. سنقوم برسم اثنين:

```
sns.regplot('Height', 'Length', X_train, fit_reg=False);
```




```
sns.regplot('Height', 'Girth', X_train, fit_reg=False);
```



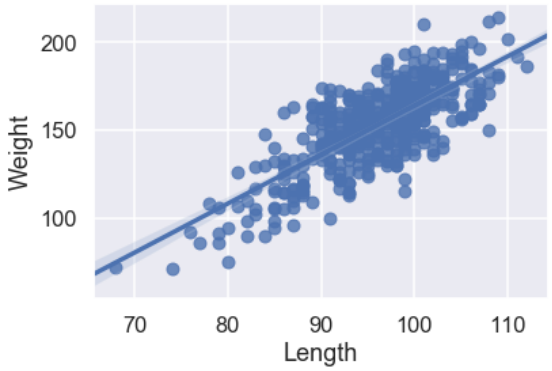
من هذه الرسوم، نلاحظ أيضاً أن المتغيرات التي تساعدنا على التنبؤ لديها علاقة خطية قوية فيما بينها. يصعب ذلك من عملية تفسير نتائج النموذج، لذا لنتذكر لذلك بعد إنشاء النموذج.

نماذج خطية أبسط

بدلاً من استخدام كل البيانات مرة واحدة، لنجرب ضبط النموذج على متغير أو اثنين أولاً.

في الأسفل ثلاث نماذج انحدار خطي فقط باستخدام متغير كمي واحد. أي هذه النماذج يبدو الأفضل؟

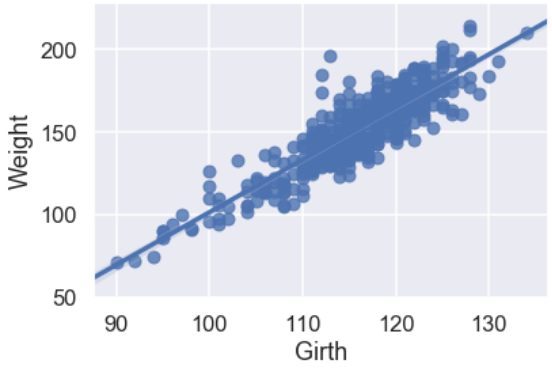
```
sns.regplot('Length', 'Weight', X_train, fit_reg=True);
```




```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train[['Length']], X_train['Weight'])
predictions = model.predict(X_test[['Length']])
print("MSE:", mse_test_set(predictions))
```

```
MSE: 26052.58007702549
```

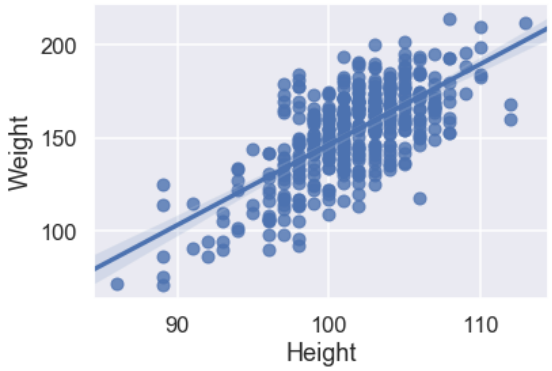
```
sns.regplot('Girth', 'Weight', X_train, fit_reg=True);
```




```
model = LinearRegression()
model.fit(X_train[['Girth']], X_train['Weight'])
predictions = model.predict(X_test[['Girth']])
print("MSE:", mse_test_set(predictions))
```

```
MSE: 13248.814105932383
```

```
sns.regplot('Height', 'Weight', X_train, fit_reg=True);
```




```
model = LinearRegression()
model.fit(X_train[['Height']], X_train['Weight'])
predictions = model.predict(X_test[['Height']])
print("MSE:", mse_test_set(predictions))
```

```
MSE: 36343.308584306134
```

بالنظر إلى مهطط التشتت ونتيجة MSE، يبدو لنا أن Girth هي الأفضل لتوقع الوزن وحدها كون لديها علاقة خطية قوية مع Weight ولديها أقل قيمة للخطأ التربيعي المتوسط.

هل سنحصل على أداء أفضل عند استخدام مُتغيرين؟ لنجرب ضبط النموذج باستخدام Girth و Length. على الرغم انه من الصعب رسم هذا النموذج، يمكننا رؤية نتيجة الخطأ التربيعي المتوسط:

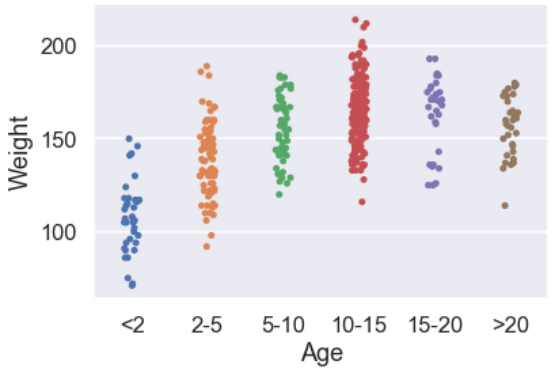
```
model = LinearRegression()
model.fit(X_train[['Girth', 'Length']], X_train['Weight'])
predictions = model.predict(X_test[['Girth', 'Length']])
print("MSE:", mse_test_set(predictions))
```

```
MSE: 9680.902423377258
```

رائع! يبدو أن MSE تم تقليلها لدينا من حوالي 13000 باستخدام Girth وحدها إلى 10000 باستخدام Girth و Length. أضافة متغير أخر حسن من نموذجنا.

يمكننا أيضاً استخدام المتغيرات الاسمية في نموذجنا. لنرى الآن كيف نستخدم النموذج الخطي مع البيانات الاسمية في العمود Age. هذا الرسم البياني ل Age و Weight:

```
sns.stripplot(x='Age', y='Weight', data=X_train, order=['<2', '2-5', '5-10', '10-15', '15-20', '>20']);
```



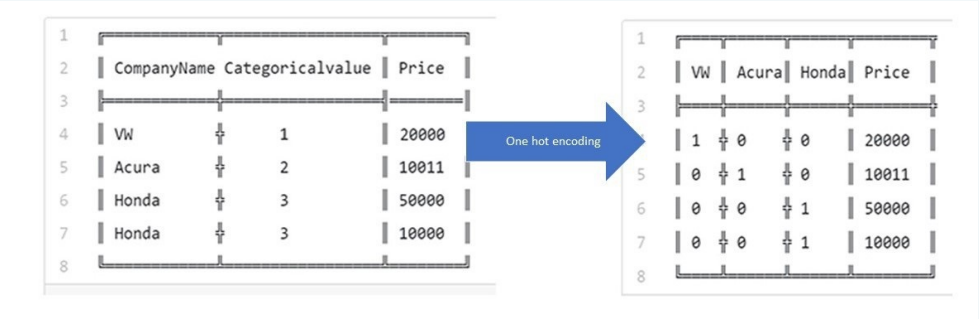
بما أن البيانات في العمود Age أسميه، فنحتاج لاستخدام بيانات رقمية مطابقة لها لنتمكن من تطبيق نموذج الانحدار الخطي عليه:

```
just_age_and_weight = X_train[['Age', 'Weight']]
with_age_dummies = pd.get_dummies(just_age_and_weight, columns=['Age'])
model = LinearRegression()
```

```
model.fit(with_age_dummies.drop('Weight', axis=1), with_age_dummies['Weight'])

just_age_and_weight_test = X_test[['Age']]
with_age_dummies_test = pd.get_dummies(just_age_and_weight_test, columns=['Age'])
predictions = model.predict(with_age_dummies_test)
print("MSE:", mse_test_set(predictions))
```

```
MSE: 41398.515625
```

استخدم الكاتب الدالة `get_dummies` لتحويل المتغيرات الاسمية إلى بيانات وهمية كمية تتكون من 0 و 1، يطلق على هذه الطريقة ب One-hot encoding ويمكن وصفها في الصورة التالية:



نتيجة 40000 أسولأ بكثير مما حصلنا عليه باستخدام متغير كمي واحد، ولكن هذا المتغير قد يثبت أهميته في نموذجنا الخطي.

لنحاول تفسير هذا النموذج. لاحظ أن أي جمار عمره لنقل بين 2 و 5 سنوات، سيحصل على نفس التوقع لأن لديهم نفس المدخلات: 1 للعمود إلى يحدد العمر بين 2-5 سنوات. و 0 في بقية الأعمدة. لذا، يمكننا تفسير المتغيرات الاسمية ببساطه بتغير القيم في النموذج لأن المتغيرات الاسمية تفصلهم إلى مجموعات وتعطي توقع واحد لجميع من في هذه المجموعة.

خطوتنا القادمة هي بناء النموذج النهائي باستخدام المتغيرات الاسمية وأكثر من متغير كمي.

تحويل المتغيرات

لنتذكر من رسمات الصندوق السابقة أن العمود `Sex` لم يكن مفيداً، لذا سنتخلص منه. سنقوم أيضاً بحذف العمود `WeightAlt` لأن لدينا فقط 31 منها. أخيراً، استخدام `get_dummies`، لتحويل البيانات الاسمية فيه `BCS` و `Age` إلى بيانات وهمية كمية لنتمكن من إدخالها إلى النموذج:

```
# هذا الخيار لزيادة عدد الأعمدة التي تظهر في جوبتر
pd.set_option('max_columns', 15)

X_train.drop(['Sex', 'WeightAlt'], axis=1, inplace=True)
X_train = pd.get_dummies(X_train, columns=['BCS', 'Age'])
X_train.head()
```

	Length	Girth	Height	BCS_1.5	BCS_2.0	BCS_2.5	BCS_3.0	BCS_3.5	BCS_4.0	Age_10-15	Age_15-20	Age_2-5	Age_5-10	Age_<2	Age_>20
465	98	113	99	0	0	0	1	0	0	0	0	1	0	0	0
233	101	119	101	0	0	0	1	0	0	1	0	0	0	0	0
450	106	125	103	0	0	1	0	0	0	1	0	0	0	0	0
453	93	120	100	0	0	1	0	0	0	0	0	1	0	0	0
452	98	120	108	0	0	1	0	0	0	0	0	0	1	0	0

لنتذكر أننا لاحظنا توزيع الوزن لمن أعمارهم أكبر من 5 وعدم وجود فارق كبير. لذا، سنجمع الأعمدة `Age_15-20`، `Age_10-15`، و `Age_>20` إلى عمود واحد:

```
age_over_10 = X_train['Age_10-15'] | X_train['Age_15-20'] | X_train['Age_>20']
X_train['Age_>10'] = age_over_10
X_train.drop(['Age_10-15', 'Age_15-20', 'Age_>20'], axis=1, inplace=True)
```

لأننا لا نريد أن تكون المصفوفة تحتوي على متغيرات كثيره، لنقوم بحذف واحد من كلا الأعمدة الاسمية `BCS` و `Age`:

```
X_train.drop(['BCS_3.0', 'Age_5-10'], axis=1, inplace=True)
X_train.head()
```

	Length	Girth	Height	BCS_1.5	BCS_2.0	BCS_2.5	BCS_3.5	BCS_4.0	Age_2-5	Age_<2	Age_>10
465	98	113	99	0	0	0	0	0	1	0	0
233	101	119	101	0	0	0	0	0	0	0	1
450	106	125	103	0	0	1	0	0	0	0	1
453	93	120	100	0	0	1	0	0	1	0	0
452	98	120	108	0	0	1	0	0	0	0	0

يجب علينا إضافة عمود جديد لإظهار الانحياز في النموذج:

```
# اضافة عمود bias
X_train = X_train.assign(bias=1)
X_train = X_train.reindex(columns=['bias'] + list(X_train.columns[:-1]))
X_train.head()
```

	bias	Length	Girth	Height	BCS_1.5	BCS_2.0	BCS_2.5	BCS_3.5	BCS_4.0	Age_2-5	Age_<2	Age_>10
465	1	98	113	99	0	0	0	0	0	1	0	0
233	1	101	119	101	0	0	0	0	0	0	0	1
450	1	106	125	103	0	0	1	0	0	0	0	1
453	1	93	120	100	0	0	1	0	0	1	0	0
452	1	98	120	108	0	0	1	0	0	0	0	0

نموذج الانحدار الخطي المتعدد

نحن الآن جاهزون لضبط النموذج باستخدام جميع المتغيرات التي قررنا أنها الأهم وبعد أن حولناها لشكل مناسب للنموذج.

يمكننا تعريف نموذجنا كالتالي:

$$f_\theta(\mathbf{x}) = \theta_0 + \theta_1(Length) + \theta_2(Girth) + \theta_3(Height) + \ldots + \theta_{11}(Age\_ > 10)$$

هذه هي الدوال التي عرفناها في درس الانحدار الخطي المتعدد، والتي سنستخدمها مرة أخرى:

```
def linear_model(thetas, X):
    '''Returns predictions by a linear model on x_vals.'''
    return X @ thetas

def mse_cost(thetas, X, y):
    return np.mean((y - linear_model(thetas, X)) ** 2)

def grad_mse_cost(thetas, X, y):
    n = len(X)
    return -2 / n * (X.T @ y  - X.T @ X @ thetas)
```

للقدرة على استخدام الدوال السابقة، نريد X و y. يمكننا أن نحصل عليهما من بياناتنا. تذكر أن X و y يجب أن تكون مصفوفات في NumPy لتتمكن من القيام بعملية الضرب باستخدام الرمز @:

```
X_train = X_train.values
y_train = y_train.values
```

الآن، فقط نحتاج لاستخدام الدالة minimize التي عرفناها في الجزء السابق:

```
thetas = minimize(mse_cost, grad_mse_cost, X_train, y_train)
```

```
theta: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] | cost: 23979.72
theta: [0.01 0.53 0.65 0.56 0.   0.   0.   0.   0.   0.   0.   0.  ] | cost: 1214.03
theta: [-0.07  1.84  2.55 -2.87 -0.02 -0.13 -0.34  0.19  0.07 -0.22 -0.3   0.43] | cost: 1002.46
theta: [-0.25 -0.76  4.81 -3.06 -0.08 -0.38 -1.11  0.61  0.24 -0.66 -0.93  1.27] | cost: 815.50
theta: [-0.44 -0.33  4.08 -2.7  -0.14 -0.61 -1.89  1.02  0.4  -1.06 -1.57  2.09] | cost: 491.91
theta: [-1.52  0.85  2.   -1.58 -0.52 -2.22 -5.63  3.29  1.42 -2.59 -5.14  5.54] | cost: 140.86
theta: [-2.25  0.9   1.72 -1.3  -0.82 -3.52 -7.25  4.64  2.16 -2.95 -7.32  6.61] | cost: 130.33
theta: [ -4.16   0.84   1.32  -0.78  -1.65  -7.09 -10.4    7.82   4.18  -3.44
 -12.61   8.24] | cost: 116.92
theta: [ -5.89   0.75   1.17  -0.5   -2.45 -10.36 -11.81  10.04   6.08  -3.6
 -16.65   8.45] | cost: 110.37
```

```
theta: [ -7.75   0.67   1.13  -0.35  -3.38 -13.76 -11.84  11.55   8.2   -3.8
 -20.     7.55] | cost: 105.74
theta: [ -9.41   0.64   1.15  -0.31  -4.26 -16.36 -10.81  11.97  10.12  -4.33
 -21.88   6.15] | cost: 102.82
theta: [-11.08   0.66   1.17  -0.32  -5.18 -18.28  -9.43  11.61  11.99  -5.37
 -22.77   4.69] | cost: 100.70
theta: [-12.59   0.69   1.16  -0.32  -6.02 -19.17  -8.53  10.86  13.54  -6.65
 -22.89   3.73] | cost: 99.34
theta: [-14.2    0.72   1.14  -0.3   -6.89 -19.35  -8.29  10.03  14.98  -7.99
 -22.74   3.14] | cost: 98.30
theta: [-16.14   0.73   1.11  -0.26  -7.94 -19.03  -8.65   9.3   16.47  -9.18
 -22.59   2.76] | cost: 97.35
theta: [-18.68   0.73   1.1   -0.21  -9.27 -18.29  -9.42   8.76  18.14 -10.04
 -22.55   2.39] | cost: 96.38

theta: [-21.93   0.72   1.1   -0.17 -10.94 -17.19 -10.25   8.5   19.92 -10.36
 -22.66   1.99] | cost: 95.35
theta: [-26.08   0.7    1.13  -0.14 -13.03 -15.78 -10.79   8.54  21.78 -10.05
 -22.83   1.59] | cost: 94.18
theta: [-31.35   0.69   1.17  -0.13 -15.59 -14.12 -10.69   8.9   23.61  -9.19
 -22.93   1.32] | cost: 92.84
theta: [-37.51   0.7    1.21  -0.13 -18.44 -12.47  -9.79   9.52  25.14  -8.06
 -22.78   1.38] | cost: 91.40
theta: [-43.57   0.72   1.23  -0.12 -21.06 -11.3   -8.4   10.2   25.98  -7.16
 -22.24   1.87] | cost: 90.06
theta: [-48.96   0.74   1.23  -0.1  -23.13 -10.82  -7.13  10.76  26.06  -6.79
 -21.34   2.6 ] | cost: 88.89
theta: [-54.87   0.76   1.22  -0.05 -25.11 -10.88  -6.25  11.22  25.55  -6.8
 -20.04   3.41] | cost: 87.62
theta: [-63.83   0.78   1.21   0.02 -27.82 -11.42  -5.83  11.68  24.36  -6.96
 -17.97   4.26] | cost: 85.79
theta: [-77.9    0.8    1.22   0.13 -31.81 -12.47  -6.17  12.03  22.29  -6.98
 -14.93   4.9 ] | cost: 83.19
theta: [-94.94   0.81   1.26   0.23 -36.3  -13.73  -7.37  11.98  19.65  -6.47
 -11.73   4.88] | cost: 80.40
theta: [-108.1     0.81    1.34    0.28  -39.34  -14.55   -8.72   11.32   17.48
   -5.47   -9.92    4.21] | cost: 78.34
theta: [-115.07    0.81    1.4     0.29  -40.38  -14.75   -9.46   10.3    16.16
   -4.47   -9.7     3.5 ] | cost: 77.07
theta: [-119.8     0.81    1.44    0.28  -40.43  -14.6    -9.61    9.02   15.09
   -3.67  -10.25    3.05] | cost: 76.03
theta: [-125.16    0.82    1.47    0.3   -40.01  -14.23   -9.3     7.48   13.79
   -3.14  -11.09    2.94] | cost: 74.96
theta: [-131.24    0.83    1.48    0.33  -39.39  -13.76   -8.71    6.21   12.41
   -3.16  -11.79    3.17] | cost: 74.03
theta: [-137.42    0.84    1.48    0.39  -38.62  -13.25   -8.11    5.57   11.18
   -3.67  -12.11    3.47] | cost: 73.23
theta: [-144.82    0.85    1.47    0.46  -37.36  -12.53   -7.56    5.47    9.93
   -4.57  -12.23    3.56] | cost: 72.28
theta: [-155.48    0.86    1.48    0.54  -34.88  -11.3    -6.98    5.95    8.38
   -5.92  -12.27    3.13] | cost: 70.91
theta: [-167.86    0.88    1.52    0.62  -31.01   -9.63   -6.53    7.03    6.9
   -7.3   -12.29    1.91] | cost: 69.33
theta: [-176.09    0.89    1.57    0.64  -27.32   -8.32   -6.41    8.07    6.31
   -7.84  -12.29    0.44] | cost: 68.19
theta: [-178.63    0.9     1.6     0.62  -25.15   -7.88   -6.5     8.52    6.6
   -7.51  -12.19   -0.39] | cost: 67.59
theta: [-179.83    0.91    1.63    0.6   -23.4    -7.84   -6.6     8.61    7.27
   -6.83  -11.89   -0.72] | cost: 67.08
theta: [-182.79    0.91    1.66    0.58  -20.55   -8.01   -6.68    8.49    8.44
   -5.7   -11.11   -0.69] | cost: 66.27
theta: [-190.23    0.93    1.68    0.6   -15.62   -8.38   -6.68    8.1    10.26
   -4.1    -9.46    0.01] | cost: 65.11
theta: [-199.13    0.93    1.69    0.67  -11.37   -8.7    -6.55    7.67   11.53
   -3.17   -7.81    1.13] | cost: 64.28
theta: [-203.85    0.93    1.68    0.72  -10.03   -8.78   -6.42    7.5    11.68
   -3.25   -7.13    1.86] | cost: 64.01
theta: [-204.24    0.93    1.67    0.74  -10.33   -8.74   -6.39    7.52   11.46
   -3.52   -7.17    1.97] | cost: 63.98
theta: [-204.06    0.93    1.67    0.74  -10.48   -8.72   -6.39    7.54   11.39
   -3.59   -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
theta: [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95] | cost: 63.98
```

ونموذجنا يبدو كالتالي:

$$y = -204.03 + 0.93x_1 + \ldots -7.22x_9 + 1.95x_{11}$$

لنقارن هذه المعادلة التي حصلنا عليها مع أخرى قد نحصل عليها من نموذج الانحدار الخطي في مكتبة `sklearn`:

```
```

```
model = LinearRegression(fit_intercept=False)
model.fit(X_train[:, :14], y_train)
print("Coefficients", model.coef_)
```

```
Coefficients [-204.03    0.93    1.67    0.74  -10.5    -8.72   -6.39    7.54   11.39
   -3.6    -7.22    1.95]
```

يظهر المعامل مطابقاً لما حصلنا عليه! دوالنا التي كتبناها أنتجت نفس النموذج كما لو أنتجته مكتبة بايثون!

تمكنا بنجاح من ضبط النموذج الخطي!

تقييم نموذجنا

خطوتنا التالية هي تقييم نتائج النموذج باستخدام بيانات الاختبار. نحتاج لتطبيق جميع ما فعلناه على بيانات التدريب في بيانات الاختبار قبل أن ندخلها إلى النموذج للتوقع:

```
X_test.drop(['Sex', 'WeightAlt'], axis=1, inplace=True)
X_test = pd.get_dummies(X_test, columns=['BCS', 'Age'])
age_over_10 = X_test['Age_10-15'] | X_test['Age_15-20'] | X_test['Age_>20']
X_test['Age_>10'] = age_over_10
X_test.drop(['Age_10-15', 'Age_15-20', 'Age_>20'], axis=1, inplace=True)
X_test.drop(['BCS_3.0', 'Age_5-10'], axis=1, inplace=True)
X_test = X_test.assign(bias=1)
X_test = X_test.reindex(columns=['bias'] + list(X_test.columns[:-1]))

X_test
```

	bias	Length	Girth	Height	BCS_1.5	BCS_2.0	BCS_2.5	BCS_3.5	BCS_4.0	Age_2-5	Age_<2	Age_>10
490	1	98	119	103	0	0	1	0	0	0	0	1
75	1	86	114	105	0	0	0	0	0	1	0	0
352	1	94	114	101	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
182	1	94	114	102	0	0	0	0	0	1	0	0
334	1	104	113	105	0	0	1	0	0	0	0	0
543	1	104	124	110	0	0	0	0	0	0	0	0

```
108 rows × 12 columns
```

نقوم بإدخال `X_test` إلى `predict` في نموذجنا الخطي `LinearRegression`:

```
X_test = X_test.values
predictions = model.predict(X_test)
```

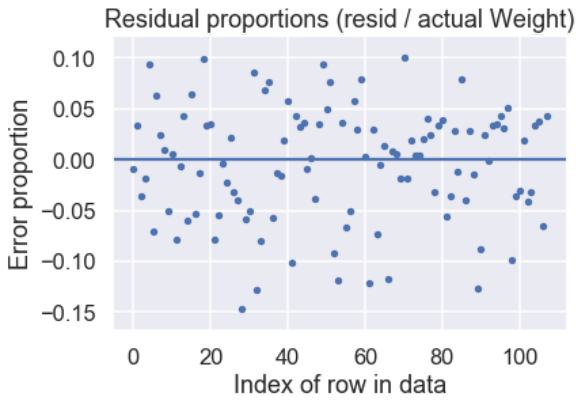
الآن لنطلع على نتيجة الخطأ التربيعي المتوسط:

```
mse_test_set(predictions)
```

```
7261.974205350604
```

بهذه التوقعات، يمكننا رسم نتيجة الفارق بين التوقع مقارنة بالنتائج الحقيقية Residual plot:

```
y_test = y_test.values
resid = y_test - predictions
resid_prop = resid / y_test
plt.scatter(np.arange(len(resid_prop)), resid_prop, s=15)
plt.axhline(0)
plt.title('Residual proportions (resid / actual Weight)')
plt.xlabel('Index of row in data')
plt.ylabel('Error proportion');
```



يبدو أن النموذج يؤدي بشكل رائع! نتيجة الفارق بين التوقع والنتائج الحقيقية تظهر أن توقعنا كان بحد أعلى 15% من النتيجة الحقيقية.

هندسة الخصائص

مقدمة

هندسة الخصائص Feature Engineering تعني إنشاء وإضافة خصائص جديده للبيانات لزيادة دقة وتعقيد النموذج.

حتى الآن، قمنا فقط بتطبيق الانحدار الخطي باستخدام خصائص كمية كمُدخلات، استخدمنا القيمة الكمية لمجموع الفاتورة للتنبؤ بمبلغ الإكرامية. ولكن، احتوت بيانات الإكراميات على بيانات اسمية، مثل أيام الأسبوع ونوع الوجبة. هندسة الخصائص تسمح لنا بتحويل البيانات الاسمية إلى كمية لاستخدامها في الانحدار الخطي.

تسمح لنا هندسة الخصائص أيضاً باستخدام نموذج الانحدار الخطي لإجراء انحدار متعدد الحدود عن طريق إنشاء متغيرات جديده في بياناتنا.

بيانات وول مارت

في عام 2014، قامت ول مارت بنشر بعض بيانات بيعها كجزء من مسابقة للتنبؤ بالمبيعات الأسبوعية في فروعها. أخذنا جزء من هذه البيانات واستخدمناها في المثال التالي:

وول مارت Walmart هي إحدى أكبر متاجر التجزئة والمنتجات اليومية في الولايات المتحدة الأمريكية.
لتحميل البيانات walmart.csv اضغط هنا.

```
import pandas as pd

walmart = pd.read_csv('walmart.csv')
walmart
```

	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	Unemployment	MarkDown
0	05-02-10	24924.5	No	42.31	2.572	8.106	No Markdown
1	12-02-10	46039.49	Yes	38.51	2.548	8.106	No Markdown
2	19-02-10	41595.55	No	39.93	2.514	8.106	No Markdown
...	...	...	...	...	...	...	...
140	12-10-12	22764.01	No	62.99	3.601	6.573	MarkDown2
141	19-10-12	24185.27	No	67.97	3.594	6.573	MarkDown2
142	26-10-12	27390.81	No	69.16	3.506	6.573	MarkDown1

```
143 rows × 7 columns
```

تحتوي البيانات على عدد من الخصائص المثيرة للإعجاب، إضافة إلى معلومات ما إذا كان الأسبوع يحتوي على إجازة في العمود IsHoliday، ومعدل البطالة في الأسبوع Unemployment وأي من العروض الخاصة التي قدمها المتجر في ذلك الأسبوع MarkDown.

الهدف هو بناء نموذج يتوقع المتغير Weekly_Sales باستخدام باقي المعلومات في جدول البيانات. باستخدام نموذج الانحدار الخطي يمكننا بشكل مباشر استخدام الأعمدة Fuel_Price، Temperature و Unemployment كونها تحتوي على بيانات كمية.

ضبط النموذج باستخدام مكتبة Scikit-learn

في الفصول السابقة تعلمنا كيف نوجد مشتقة دالة التكلفة واستخدام النزول الاشتقاقي لضبط النموذج. لفعل ذلك، كان علينا تعريف دوال في بايثون لبناء النموذج، دالة التكلفة، مشتقة دالة التكلفة، وخوارزمية النزول الاشتقاقي. رغم أهمية ذلك لعرض وفهم المفاهيم في بناء النماذج، في هذا الفصل سنستخدم مكتبة متخصصة بتعلم الآلة اسمها `scikit-learn` والتي تسمح لنا بضبط النماذج باستخدام أكواد برمجية أقل.

مثلاً، لضبط نموذج انحدار خطي متعدد باستخدام البيانات الكمية في قاعدة بيانات وول مارت، نقوم أولاً ببناء مصفوفة ثنائية الأبعاد باستخدام NumPy تحتوي على المتغيرات المستخدمة لبناء نموذج التنبؤ و مصفوفة أحادية الأبعاد تحتوي على القيم التي نريد التنبؤ عنها:

```
numerical_columns = ['Temperature', 'Fuel_Price', 'Unemployment']
X = walmart[numerical_columns].to_numpy()
X
```

```
array([[ 42.31,   2.57,   8.11],
       [ 38.51,   2.55,   8.11],
       [ 39.93,   2.51,   8.11],
       ...,
       [ 62.99,   3.6 ,   6.57],
       [ 67.97,   3.59,   6.57],
       [ 69.16,   3.51,   6.57]])
```

```
y = walmart['Weekly_Sales'].to_numpy()
y
```

```
array([ 24924.5 ,  46039.49,  41595.55, ...,  22764.01,  24185.27,
        27390.81])
```

ثم نستدعي كلاس الانحدار الخطي `LinearRegression` من مكتبة `scikit-learn` (مزيد من التفاصيل اضغط هنا)، نعرّف بالنموذج، ثم نستدعي دالة الضبط `fit` باستخدام المتغيرات `X` للتنبؤ بقيمة `y`.

لاحظ أننا في السابق احتجنا لإضافة عمود جديد يدوياً يحتوي على القيمة 1 للمصفوفة `X` "التحيز" لنقوم بتطبيق الانحدار الخطي. هذه المرة، `scikit-learn` ستتكفل بفعل ذلك لتختصر علينا الوقت:

```
from sklearn.linear_model import LinearRegression

simple_classifier = LinearRegression()
simple_classifier.fit(X, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

رائع! عندما نستدعي `.fit`، مكتبة `scikit-learn` أوجدت المتغيرات التي تقلل من دالة تكلفة المربعات الصغرى. يمكننا الاطلاع على المتغيرات كالتالي:

```
simple_classifier.coef_, simple_classifier.intercept_
```

```
(array([ -332.22,  1626.63,  1356.87]), 29642.700510138635)
```

لحساب تكلفة المتوسط التربيعي، يمكننا أن نطلب من المصنف Classifier التنبؤ للقيم المدخلة في `X` ومقارنة النتيجة مع البيانات الحقيقة في `y`:

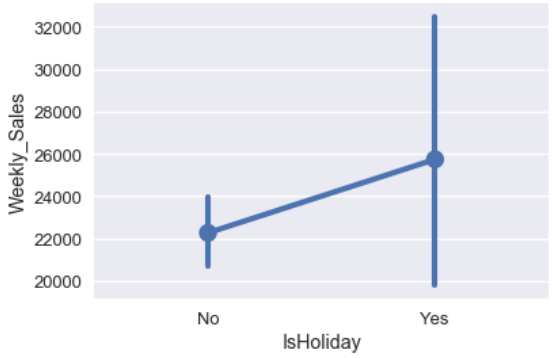
```
import numpy as np

predictions = simple_classifier.predict(X)
np.mean((predictions - y) ** 2)
```

```
74401210.603607252
```

يظهر أن الخطأ التربيعي المتوسط يبدو مرتفعاً جداً. على الأرجح يكون سبب ذلك هو المتغيرات الثلاثة الكمية التي استخدمناها وارتباطها الضعيف بالمبيعات الأسبوعية.

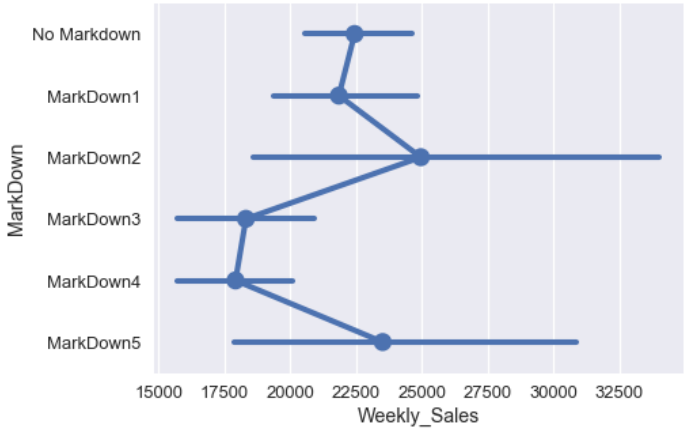
لدينا متغيران آخران من الممكن أن يكون لهما فائدة في التنبؤ: عمود `IsHoliday` و `MarkDown`. مخطط الصندوق في الأسفل يوضح أن الإجازات قد يكون لها ارتباط مع المبيعات الأسبوعية:

```
import seaborn as sns

sns.pointplot(x='IsHoliday', y='Weekly_Sales', data=walmart);
```



يظهر ان هناك رابط بين الأنواع المختلفة للعروض في العمود `MarkDown` وبين كمية المبيعات في الأسابيع المختلفة:

```
import matplotlib.pyplot as plt

markdowns = ['No Markdown', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']
plt.figure(figsize=(7, 5))
sns.pointplot(x='Weekly_Sales', y='MarkDown', data=walmart, order=markdowns);
```



ولكن، كلا الأعمدة `IsHoliday` و `MarkDown` هي بيانات أسميه، وليست كمية، لذا لا يمكن أن نستخدمها كما هي الآن في الانحدار.

استخدام One-Hot Encoding

لحسن الحظ، يمكننا أن نستخدم **One-Hot Encoding** على هذه البيانات الاسمية لتحويلها لكمية. التحويل يتم كالتالي: ننشأ عمود جديد لكل قيمه مختلفة في العمود الأسمي. يحتوي كل عمود على الرقم 1 إذا كانت العمود هو تلك القيمة الموجودة في العمود الأسمي، وغير ذاك تكون القيمة 0، مثلاً في العمود `MarkDown`:

```
walmart[['MarkDown']]
```

	MarkDown
0	No Markdown
1	No Markdown
2	No Markdown
...	...
140	MarkDown2
141	MarkDown2

	MarkDown
142	MarkDown1

```
143 rows × 1 columns
```

يحتوي العمود على ست قيم مختلفة وهي: MarkDown4، MarkDown3، MarkDown2، MarkDown1، No Markdown، و MarkDown5. نقوم بإنشاء عمود لكل قيمه وينتج لنا ست أعمدة جديده. ثم نقوم بتعبئة كل عمود بصفر أو واحد بنفس الطريقة التي شرحناها مسبقاً.

```
from sklearn.feature_extraction import DictVectorizer

items = walmart[['MarkDown']].to_dict(orient='records')
encoder = DictVectorizer(sparse=False)
pd.DataFrame(
    data=encoder.fit_transform(items),
    columns=encoder.feature_names_
)
```

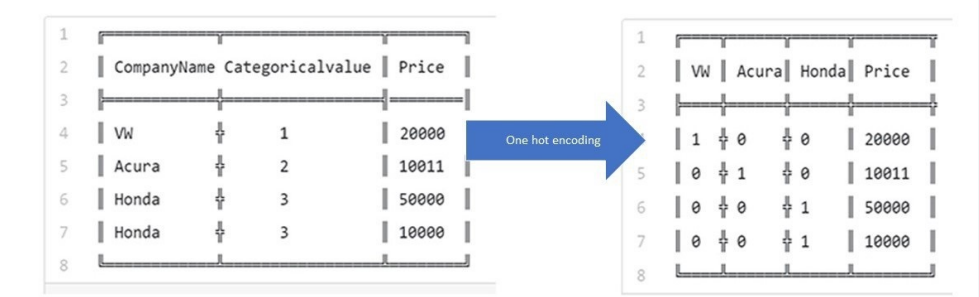
	MarkDown=MarkDown1	MarkDown=MarkDown2	MarkDown=MarkDown3	MarkDown=MarkDown4	MarkDown=MarkDown5	MarkDown=No Markdown
0	0	0	0	0	0	1
1	0	0	0	0	0	1
2	0	0	0	0	0	1
...	...	...	...	...	...	...
140	0	1	0	0	0	0
141	0	1	0	0	0	0
142	1	0	0	0	0	0

```
143 rows × 6 columns
```

لاحظ أن أول سطر في البيانات هي No Markdown، لذا فقط آخر عمود في الجدول الذي تم إنشاءه تحتوي على الرقم 1. وأيضاً، آخر سطر في البيانات يحتوي على MarkDown1 مما جعل أو عمود يحتوي على 1.

كل سطر في الجدول يحتوي على عمود واحد لدية الرقم 1، والبقية ستحتوي على 0. السم "One-Hot" يعني أن عمود واحد سيكون ذو قيمه (يحتوي على 1).

مثال آخر تم ذكره في الفصل السابق:



طريقة One-Hot Encoding في مكتبة Scikit-learn

لنقوم بعملية One-Hot Encoding يمكننا استخدام الكلاس DictVectorizer من مكتبة scikit-learn. لنستخدم هذه الكلاس، نحتاج لتحويل ال DataFrame إلى مصفوفة تحتوي على قواميس. الكلاس DictVectorizer يقوم بشكل تلقائي بعملية One-Hot Encoding للأعمدة الاسمية (والتي يجب أن تكون نصوص) ولا يقوم بتغير أي عمود يحتوي على قيم كمية:

```
from sklearn.feature_extraction import DictVectorizer

all_columns = ['Temperature', 'Fuel_Price', 'Unemployment', 'IsHoliday',
               'MarkDown']

records = walmart[all_columns].to_dict(orient='records')
encoder = DictVectorizer(sparse=False)
encoded_X = encoder.fit_transform(records)
encoded_X
```

```
array([[  2.57,   1.  ,   0.  , ...,   1.  ,  42.31,   8.11],
       [  2.55,   0.  ,   1.  , ...,   1.  ,  38.51,   8.11],
       [  2.51,   1.  ,   0.  , ...,   1.  ,  39.93,   8.11],
       ...,
       [  3.6 ,   1.  ,   0.  , ...,   0.  ,  62.99,   6.57],
       [  3.59,   1.  ,   0.  , ...,   0.  ,  67.97,   6.57],
       [  3.51,   1.  ,   0.  , ...,   0.  ,  69.16,   6.57]])
```

لتتبين لك الفكرة والشكل الجديد للبيانات، يمكننا عرضها مع أسماء الأعمدة:

```
pd.DataFrame(data=encoded_X, columns=encoder.feature_names_)
```

	Fuel_Price	IsHoliday=No	IsHoliday=Yes	MarkDown=MarkDown1	...	MarkDown=MarkDown5	MarkDown=No Markdown	Temperature	Unemployment
0	2.572	1	0	0	...	0	1	42.31	8.106
1	2.548	0	1	0	...	0	1	38.51	8.106
2	2.514	1	0	0	...	0	1	39.93	8.106
...	...	...	...	...	...	...	...	...	...
140	3.601	1	0	0	...	0	0	62.99	6.573
141	3.594	1	0	0	...	0	0	67.97	6.573
142	3.506	1	0	1	...	0	0	69.16	6.573

143 rows × 11 columns

الأعمدة الكمية Temperature، Fuel price و Unemployment بقيت كما هي كأرقام. الأعمدة الاسمية IsHoliday و MarkDown تم تطبيق ال One-Hot Encoding عليها. عندما نستخدم المصفوفة الجديدة للبيانات لضبط نموذج الانحدار الخطي، سنقوم بإيجاد متغير جديد لكل عمود في البيانات. بما أن المصفوفة تحتوي على 11 أعمدة، النموذج سيتنبأ ب 12 متغير بما أننا أضفنا متغير إضافي للتحيز.

ضبط النموذج باستخدام البيانات المُعدلة

يمكننا الآن استخدام encoded_X في نموذج الانحدار الخطي:

```
clf = LinearRegression()
clf.fit(encoded_X, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

كما ذكرنا مُسبقاً، ستكون النتيجة مكونة من 12 قيمه لكل عمود:

```
clf.coef_, clf.intercept_
```

```
(array([ 1622.11,    -2.04,     2.04,   962.91,  1805.06, -1748.48,
        -2336.8 ,   215.06,  1102.25,  -330.91,  1205.56]), 29723.135729284979)
```

يمكننا مقارنة نتائج التوقع بين كلا النماذج لنرى إذا كان هناك فرق كبير بينهما:

```
walmart[['Weekly_Sales']].assign(
    pred_numeric=simple_classifier.predict(X),
    pred_both=clf.predict(encoded_X)
)
```

	Weekly_Sales	pred_numeric	pred_both
0	24924.5	30768.87804	30766.79021
1	46039.49	31992.2795	31989.4104
2	41595.55	31465.22016	31460.28001

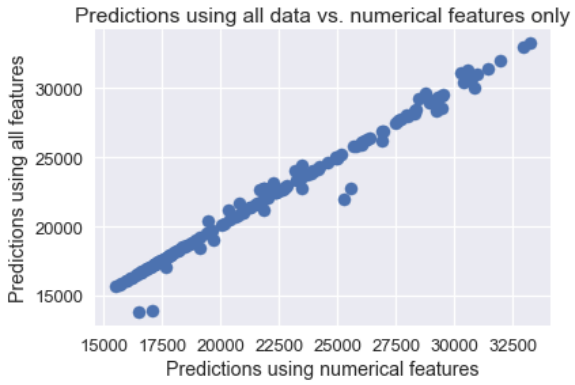
	Weekly_Sales	pred_numeric	pred_both
...	...	...	...
140	22764.01	23492.26265	24447.34898
141	24185.27	21826.41479	22788.04955
142	27390.81	21287.92854	21409.36746

```
143 rows × 3 columns
```

في الجدول السابق، العمود `pred_numeric` يمثل نتائج النموذج `simple_classifier` والذي استخدم فقط الأعمدة الكمية في بياناتنا. بينما `pred_both` استخدم كامل الأعمدة.

نلاحظ أن نتائج التوقع قريبه بين النموذجان. يمكن رسم مخطط التشتت لكلا الأعمدة لتوضيح ذلك:

```
plt.scatter(simple_classifier.predict(X), clf.predict(encoded_X))
plt.title('Predictions using all data vs. numerical features only')
plt.xlabel('Predictions using numerical features')
plt.ylabel('Predictions using all features');
```



تقييم النموذج

لماذا ظهرت لنا النتائج هكذا؟ يمكننا عرض المتغيرات التي اعتمدها النموذجان. الجدول في الأسفل يوضح الأوزان المناسبة التي تعلم عليها المُصنف باستخدام الأعمدة الكمية فقط:

```
def clf_params(names, clf):
    weights = (
        np.append(clf.coef_, clf.intercept_)
    )
    return pd.DataFrame(weights, names + ['Intercept'])

clf_params(numerical_columns, simple_classifier)
```

	0
Temperature	332.22118-
Fuel_Price	1626.625604
Unemployment	1356.868319
Intercept	29642.70051

الجدول في الأسفل يوضح الأوزان المناسبة التي تعلم عليها المُصنف بعد تطبيق One-Hot Encoding على البيانات:

```
pd.options.display.max_rows = 13
display(clf_params(encoder.feature_names_, clf))
pd.options.display.max_rows = 7
```

	0
Fuel_Price	1622.106239
IsHoliday=No	2.041451-

	0
IsHoliday=Yes	2.041451
MarkDown=MarkDown1	962.908849
MarkDown=MarkDown2	1805.059613
MarkDown=MarkDown3	1748.475046-
MarkDown=MarkDown4	2336.799791-
MarkDown=MarkDown5	215.060616
MarkDown=No Markdown	1102.24576
Temperature	330.912587-
Unemployment	1205.564331
Intercept	29723.13573

في الكود البرمجي السابق، عرف الكاتب دالة `clf_params` والتي تستقبل اسم الأعمدة و النموذج، وتنتج لنا DataFrame بقيم الأوزان لكل عمود. العمود `Intercept` هنا هو الانحياز.

نلاحظ أنه حتى بعد ضبط نموذج الانحدار الخطي باستخدام One-Hot Encoding الأوزان للأعمدة `Temperature`، `Fuel price` و `Unemployment` مشابهة بشكل كبير للبيانات قبل التعديل. جميع قيم الأوزان قليلة مقارنة بالانحياز `Intercept`، يشير ذلك إلى أن أكثر المتغيرات لا يزال ارتباطها قليل مع القيم الحقيقة للمبيعات. في الحقيقة، وزن العمود `IsHoliday` في النموذج قليل جداً مما يجعله لا يشكل فرقاً في التنبؤ. على الرغم أن بعض الأوزان في العمود `MarkDown` تبدو مرتفعه جداً، الكثير منها لم يظهر إلا قليلاً:

```
walmart['MarkDown'].value_counts()
```

```
No Markdown    92
MarkDown1      25
MarkDown2      13
MarkDown5       9
MarkDown4       2
MarkDown3       2
Name: MarkDown, dtype: int64
```

يشير ذلك أننا نحتاج لجمع المزيد من البيانات ليكون للعمود `MarkDown` تأثير على قيمة المبيعات. (في الحقيقة، البيانات هنا هي جزء صغير من البيانات الحقيقة الكاملة والتي نشرتها وول مارت. سيكون تدريباً مناسباً لو قمت بتجربة ضبط النموذج وتدريبه على كامل البيانات بدلاً من جزء بسيط منها).

ملخص بيانات وول مارت

تعلمنا كيف نستخدم One-Hot Encoding، طريقة مناسبة لتطبيق الانحدار الخطي على البيانات الاسمية. على الرغم أن في هذا المثال تطبيق ذلك لم يؤثر كثيراً على النموذج، عملياً تستخدم هذه الطريقة بشكل كبير للتعامل مع البيانات الاسمية. One-Hot Encoding تعتبر أحد المبادئ العامة في هندسة الخصائص، تأخذ مصفوفة/عمود من البيانات وتحولها إلى مصفوفات/أعمدة ذات أهمية أكبر.

التنبؤ بتقييم الآيس كريم

لنفترض أننا نريد صناعة نكهة جديده وتكسب شهره من الآيس كريم. نحن مهتمون بحل مشكلة الانحدار التالية: بناءًا على نسبة حلاوة نكهة الآيس كريم، نريد التنبؤ عن التقييم العام للآيس كريم من 7.

لتحميل البيانات icecream.csv اضغط هنا.

```
ice = pd.read_csv('icecream.csv')
ice
```

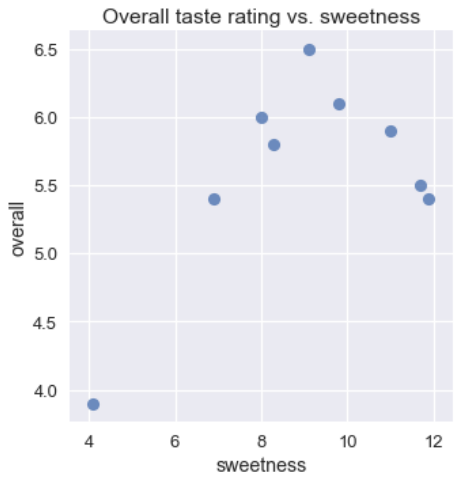
	sweetness	overall
0	4.1	3.9
1	6.9	5.4
2	8.3	5.8
...	...	...
6	11	5.9

	sweetness	overall
7	11.7	5.5
8	11.9	5.4

```
9 rows × 2 columns
```

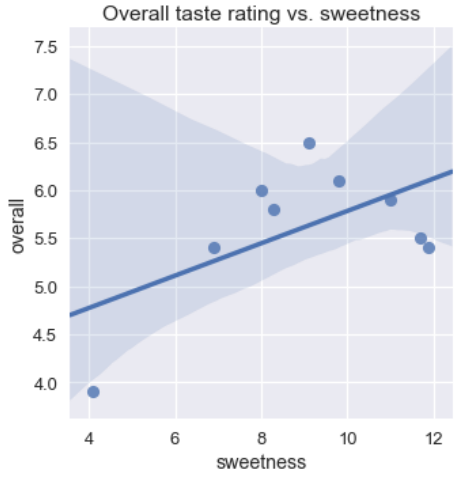
على الرغم من توقعنا أن نكهات الآيس كريم التي لا تعتبر ذات حلاوة عاليه ستحصل على تقييم أقل، نتوقع أيضاً أن النكهات ذات الحلاوة العالية ستحصل أيضاً على تقييم أقل. يتبين لنا ذلك في رسمه مخطط التشتت للتقييم مقارنة بحلاوة الآيس كريم:

```
sns.lmplot(x='sweetness', y='overall', data=ice, fit_reg=False)
plt.title('Overall taste rating vs. sweetness');
```



للأسف، لا يمكن للنموذج الخطي وحدة من التعامل مع الزيادة والنقصان الواضحة في الرسم البياني؛ في النموذج الخطي، التقييم يزداد أو ينقص مع حلاوة الآيس كريم. نرى أن استخدام النموذج الخطي يظهر لنا نتائج سيئة.

```
sns.lmplot(x='sweetness', y='overall', data=ice)
plt.title('Overall taste rating vs. sweetness');
```



طريقة مفيدة لحل مثل هذه المشاكل هي باستخدام المنحنيات متعددة الحدود بدلاً من الخط. مثل هذه المنحنيات تساعدنا على بناء نموذج يتعامل مع الزيادة في التقييم حتى الوصول لنقطة معينة من الحلاوة، ثم النقص بالتقييم مع الزيادة في الحلاوة.

باستخدام تقنيات معينة في هندسة الخصائص، يمكننا أن ننشأ أعمدة جديده في البيانات لتساعدنا على استخدام النموذج الخطي للانحدار متعدد الخطوط.

خصائص متعددة الحدود

لنتذكر أن في الانحدار الخطي نقوم بضبط وزن واحد لكل عمود في المصفوفة X. في هذه الحالة، في المصفوفة X سيكون لدينا عامودان: عمود يحتوي على الرقم 1 و آخر يحتوي على مستوى حلاوة الآيس كريم:

```
from sklearn.preprocessing import PolynomialFeatures

first_X = PolynomialFeatures(degree=1).fit_transform(ice[['sweetness']])
pd.DataFrame(data=first_X, columns=['bias', 'sweetness'])
```

	bias	sweetness
0	1.0	4.1
1	1.0	6.9
2	1.0	8.3
...	...	...
6	1.0	11
7	1.0	11.7
8	1.0	11.9

```
9 rows × 2 columns
```

بذلك يمكننا تعريف النموذج كالتالي:

$$f_{\hat{\theta}}(x) = \hat{\theta_0} + \hat{\theta_1} \cdot \text{sweetness}$$

يمكننا إنشاء عمود جديد في X يحتوي على تربيع العمود sweetness:

```
second_X = PolynomialFeatures(degree=2).fit_transform(ice[['sweetness']])
pd.DataFrame(data=second_X, columns=['bias', 'sweetness', 'sweetness^2'])
```

	bias	sweetness	sweetness^2
0	1.0	4.1	16.81
1	1.0	6.9	47.61
2	1.0	8.3	68.89
...	...	...	...
6	1.0	11	121
7	1.0	11.7	136.89
8	1.0	11.9	141.61

```
9 rows × 3 columns
```

بما أن النموذج سيتعلم الحصول على وزن واحد لكل عمود في المصفوفة المعطاة له، سيكون نموذجنا كالتالي:

$$f_{\hat{\theta}}(x) = \hat{\theta_0} + \hat{\theta_1} \cdot \text{sweetness} + \hat{\theta_2} \cdot \text{sweetness}^2$$

يمكن لنموذجنا الآن أن يضبط متعددة الخطوط من الدرجة الثانية. يمكننا أن نضبط درجات أعلى من متعددة الخطوط بإضافة أعمدة جديده لحلاوة الآيس كريم sweetness^3 ، sweetness^4 وهكذا.

لاحظ أن هذا النموذج لا يزال نموذج خطي، **لأن مُعلماته خطية**، كل $\hat{\theta_i}$ هي قيمة عددية من الدرجة الأولى. ولكن، النموذج **متعدد الحدود بخصائصه** لأن البيانات تحتوي على عمود متعدد الحدود تم إيجاده من عمود آخر.

الانحدار متعدد الحدود

لتطبيق الانحدار متعدد الحدود، نستخدم نموذج خطي مع خصائص متعددة الحدود. لذا، نقوم بإستدعاء النموذج `LinearRegression` و `PolynomialFeatures` من مكتبة `scikit-learn`.

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
```

تحتوي المصفوفة الأصليه X على القيم التاليه. تذكر أن البيانات الأصلية لا تحتوي على الرقم التسلسلي ومسمى العواميد، المصفوفة الأصلية X تحتوي على الأرقام فقط:

```
ice[['sweetness']]
```

	sweetness
0	4.1
1	6.9
2	8.3
...	...
6	11
7	11.7
8	11.9

```
9 rows × 1 columns
```

نستخدم أولاً الكلاس `PolynomialFeatures` لتحويل البيانات، نضيف الخصائص متعددة الحدود من الدرجة الثانية:

```
transformer = PolynomialFeatures(degree=2)
X = transformer.fit_transform(ice[['sweetness']])
X
```

```
array([[  1.  ,   4.1 ,  16.81],
       [  1.  ,   6.9 ,  47.61],
       [  1.  ,   8.3 ,  68.89],
       ...,
       [  1.  ,  11.  , 121.  ],
       [  1.  ,  11.7 , 136.89],
       [  1.  ,  11.9 , 141.61]])
```

الآن، نقوم بضبط النموذج الخطي على بيانات هذه المصفوفة:

```
clf = LinearRegression(fit_intercept=False)
clf.fit(X, ice['overall'])
clf.coef_
```

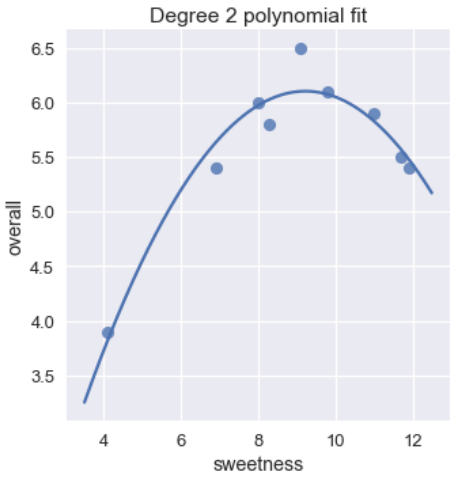
```
array([-1.3 ,  1.6 , -0.09])
```

المتغيرات السابقة تظهر أن لهذه البيانات، أفضل قيمة لضبط النموذج هي:

$$f_{\hat{\theta}}(x) = -1.3 + 1.6 \cdot \text{sweetness} - 0.09 \cdot \text{sweetness}^2$$

الآن، يمكننا مقارنة توقعات هذا النموذج مع البيانات الأصلية:

```
sns.lmplot(x='sweetness', y='overall', data=ice, fit_reg=False)
xs = np.linspace(3.5, 12.5, 1000).reshape(-1, 1)
ys = clf.predict(transformer.transform(xs))
plt.plot(xs, ys)
plt.title('Degree 2 polynomial fit');
```



هذا النموذج يظهر أنه أفضل بكثير من النموذج الخطي، يمكننا التأكد أيضاً من أن تكلفة المتوسط التربيعي لمتعددة الحدود من الدرجة الثانية أقل بكثير من التكلفة للخطي:

```
y = ice['overall']
pred_linear = (
    LinearRegression(fit_intercept=False).fit(first_X, y).predict(first_X)
)
pred_quad = clf.predict(X)

# دالة لحاسب المتوسط
def mse_cost(pred, y):
    return np.mean((pred - y) ** 2)

print(f'MSE cost for linear reg:     {mse_cost(pred_linear, y):.3f}')
print(f'MSE cost for deg 2 poly reg: {mse_cost(pred_quad, y):.3f}')
```

```
MSE cost for linear reg:     0.323
MSE cost for deg 2 poly reg: 0.032
```

زيادة الدرجة

كما ذكرنا سابقاً، يمكننا زيادة درجة متعددة الحدود بإضافة المزيد من الخصائص للبيانات. مثلاً، يمكننا بسهولة إنشاء خصائص من الدرجة الخامسة لمتعددة الحدود وستكون كالتالي:

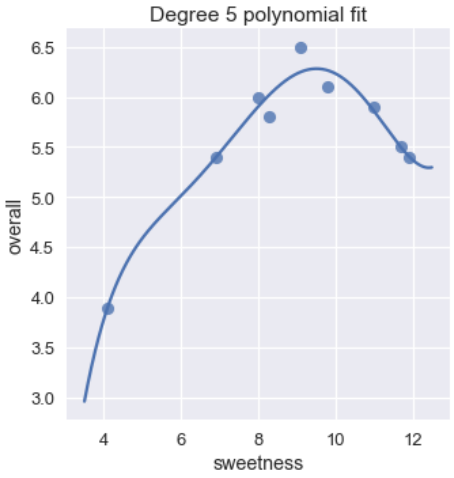
```
second_X = PolynomialFeatures(degree=5).fit_transform(ice[['sweetness']])
pd.DataFrame(data=second_X,
             columns=['bias', 'sweetness', 'sweetness^2', 'sweetness^3',
                      'sweetness^4', 'sweetness^5'])
```

	bias	sweetness	sweetness^2	sweetness^3	sweetness^4	sweetness^5
0	1	4.1	16.81	68.921	282.5761	1158.56201
1	1	6.9	47.61	328.509	2266.7121	15640.31349
2	1	8.3	68.89	571.787	4745.8321	39390.40643
...	...	...	...	...	...	...
6	1	11	121	1331	14641	161051
7	1	11.7	136.89	1601.613	18738.8721	219244.8036
8	1	11.9	141.61	1685.159	20053.3921	238635.366

9 rows × 6 columns

ضبط النموذج الخطي باستخدام هذه الخصائص سينتج لنا الانحدار متعدد الحدود من الدرجة الخامسة:

```
trans_five = PolynomialFeatures(degree=5)
X_five = trans_five.fit_transform(ice[['sweetness']])
clf_five = LinearRegression(fit_intercept=False).fit(X_five, y)

sns.lmplot(x='sweetness', y='overall', data=ice, fit_reg=False)
xs = np.linspace(3.5, 12.5, 1000).reshape(-1, 1)
ys = clf_five.predict(trans_five.transform(xs))
plt.plot(xs, ys)
plt.title('Degree 5 polynomial fit');
```



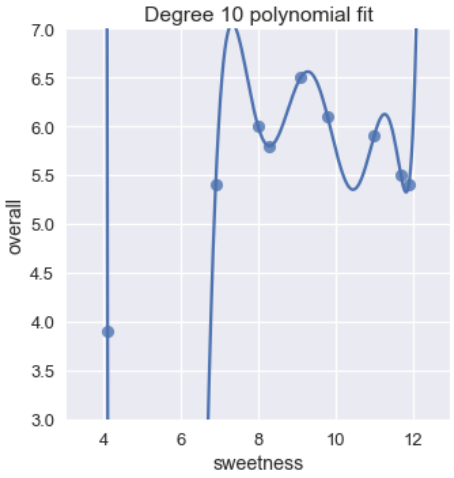
الرسم البياني يظهر أن متعددة الحدود من الدرجة الخامسة نتائجها مشابهة بشكل كبير لمتعددة الحدود من الدرجة الثانية. في الحقيقة، تكلفة المتوسط التربيعي لمتعددة الحدود ذات الدرجة الخامسة بلغت نصف تكلفة المتوسط التربيعي لمتعددة الحدود من الدرجة الثانية:

```
pred_five = clf_five.predict(X_five)

print(f'MSE cost for linear reg:     {mse_cost(pred_linear, y):.3f}')
print(f'MSE cost for deg 2 poly reg: {mse_cost(pred_quad, y):.3f}')
print(f'MSE cost for deg 5 poly reg: {mse_cost(pred_five, y):.3f}')
```

```
MSE cost for linear reg:     0.323
MSE cost for deg 2 poly reg: 0.032
MSE cost for deg 5 poly reg: 0.017
```

يعني ذلك أننا قد نحصل على نتائج أفضل لو قمنا بزايدة الدرجة. لماذا لا نجرب متعددة الحدود من الدرجة العاشرة؟

```
trans_ten = PolynomialFeatures(degree=10)
X_ten = trans_ten.fit_transform(ice[['sweetness']])
clf_ten = LinearRegression(fit_intercept=False).fit(X_ten, y)

sns.lmplot(x='sweetness', y='overall', data=ice, fit_reg=False)
xs = np.linspace(3.5, 12.5, 1000).reshape(-1, 1)
ys = clf_ten.predict(trans_ten.transform(xs))
plt.plot(xs, ys)
plt.title('Degree 10 polynomial fit')
plt.ylim(3, 7);
```



نتائج تكلفة المتوسط التربيعي لجميع نماذج الانحدار التي عرضناها:

```
pred_ten = clf_ten.predict(X_ten)

print(f'MSE cost for linear reg:     {mse_cost(pred_linear, y):.3f}')
```

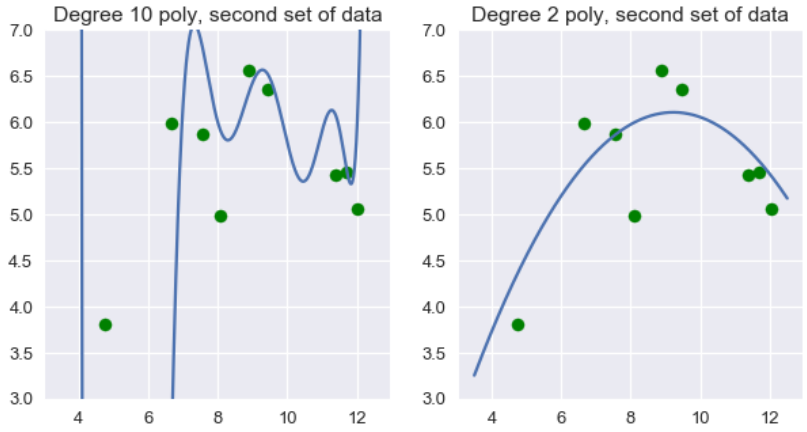
```
print(f'MSE cost for deg 2 poly reg:  {mse_cost(pred_quad, y):.3f}')
print(f'MSE cost for deg 5 poly reg:  {mse_cost(pred_five, y):.3f}')
print(f'MSE cost for deg 10 poly reg: {mse_cost(pred_ten, y):.3f}')
```

```
MSE cost for linear reg:      0.323
MSE cost for deg 2 poly reg:  0.032
MSE cost for deg 5 poly reg:  0.017
MSE cost for deg 10 poly reg: 0.000
```

متعددة الحدود من الدرجة العاشرة تكلفتها كانت صفرا! سنفهم ذلك إذا أمعنا النظر في الرسم البياني؛ تمكنت متعددة الحدود من الدرجة العاشرة من المرور بشكل صحيح على مكان كل نقطة في البيانات.

ولكن، يجب أن تتردد من استخدام الدرجة العاشرة من متعددة الحدود لتوقع تقييم الآيس كريم. متعددة الحدود من الدرجة العاشرة تظهر أنها مضبوطة للبيانات التي لدينا بشكل متقن. إذا حصلنا على بيانات جديده ورسمنا النتائج على مخطط التشتت، نتوقع أن تكون النتائج قريبة من البيانات الأصلية. لو جربنا ذلك على متعددة الحدود من الدرجة العاشرة، فإن النتائج تظهر أسوأ بكثير من متعددة الحدود من الدرجة الثانية:

```
# انشاء بيانات عشوائية ذات حجم مشابه
# للبيانات الأصليه في المصفوفة ice
# https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
np.random.seed(1)
x_devs = np.random.normal(scale=0.4, size=len(ice))
y_devs = np.random.normal(scale=0.4, size=len(ice))

plt.figure(figsize=(10, 5))

# الدرجة 10 على البيانات الجديده
plt.subplot(121)
ys = clf_ten.predict(trans_ten.transform(xs))
plt.plot(xs, ys)
plt.scatter(ice['sweetness'] + x_devs,
            ice['overall'] + y_devs,
            c='g')
plt.title('Degree 10 poly, second set of data')
plt.ylim(3, 7);

# الدرجة 2 على البيانات الجديده
plt.subplot(122)
ys = clf.predict(transformer.transform(xs))
plt.plot(xs, ys)
plt.scatter(ice['sweetness'] + x_devs,
            ice['overall'] + y_devs,
            c='g')
plt.title('Degree 2 poly, second set of data')
plt.ylim(3, 7);
```



نرى في هذه الحالة، أن متعددة الحدود من الدرجة الثانية حصلت على نتائج أفضل من النموذج دون تغير في الخصائص و متعددة الحدود من الدرجة العاشرة.

ذلك يجعلنا نتسائل: بشكل عام، متى نحدد الضبط المناسب لدرجة متعددة الحدود؟ على الرغم من ترددنا من استخدام التكلفة على بيانات التدريب لإختيار أفضل متعددة الحدود، رأينا أن استخدام التكلفة قد يجعلنا نختار نماذج معقدة جداً. بدلاً من ذلك، نريد تقييم النموذج على بيانات لم تستخدم في ضبط متغيراته.

ملخص التنبؤ بتقييم الآيس كريم

في هذا الجزء، تعرفنا على طريقة أخرى لهندسة الخصائص: إضافة خصائص متعددة الحدود للبيانات من أجل تطبيق الانحدار متعدد الحدود. كما في One-Hot Encoding، إضافة خصائص متعددة الحدود يسمح لنا باستخدام نموذج الانحدار الخطي بشكل فعال على أنواع متعددة من البيانات.

رأينا أيضاً مشكلة أساسية تواجهنا عندما نقوم بهندسة الخصائص. إضافة الكثير من الخصائص يعطي نموذجنا تكلفة أقل على البيانات الأصلية ولكن عادة ما يعطي نتائج غير صحيحه عند تطبيق النموذج على بيانات جديدة.

المقايضة بين الانحياز والتباين

مقدمة

في بعض الأحيان، نختار نموذج بسيط جداً ليمثل البيانات. وفي بعض المرات، نختار نموذجاً معقد، بسبب ضبطنا على الضوضاء في البيانات بدلاً من البيانات نفسها.

لفهم سبب قيامنا بذلك، نقوم بتحليل النموذج باستخدام أدوات الاحتمالات والإحصاء. هذه الأدوات تسمح لنا بالتعميم من أمثله بسيطة محدودة حتى وصف المهام الأساسية في النمذجة. بشكل خاص، سنستخدم **التوقع Expectation** و **التباين Variance** لعرض وفهم المقايضة بين الانحياز والتباين.

تقليل المخاطر والخسائر

للقيام بالتوقع باستخدام البيانات، نقوم بتعريف نموذج، اختيار دالة خسارة لجميع البيانات، وضبط النموذج مع المتغيرات عن طريق تقليل الخسارة. مثلاً، للقيام بالانحدار الخطي للمربعات الصغرى، نختار النموذج:

$$f_{\hat{\theta}}(x) = \hat{\theta} \cdot x$$

ودالة الخسارة:

$$L(\hat{\theta}, X, y) = \frac{1}{n} \sum_i (y_i - f_\theta(X_i))^2$$

كما فعلنا مسبقاً، نستخدم $\hat{\theta}$ كمصفوفة لمتغيرات النموذج، و x متّجه تحتوي على سطر من مصفوفة البيانات X، و y هي مصفوفة للبيانات التي تعلم عليها لتساعد على التوقع. X_i هي السطر i من المصفوفة X و y_i هي النتيجة رقم i في المصفوفة y.

لاحظ أن الخسارة لكامل البيانات هي متوسط نتائج دالة الخسارة لكل سطر فيها. إذا قمنا بتعريف دالة الخسارة التربيعية:

$$\ell(y_i, f_{\hat{\theta}}(x)) = (y_i - f_{\hat{\theta}}(x))^2$$

إذاً، يمكننا إعادة كتابة متوسط الخسارة بشكل أبسط:

$$L(\hat{\theta}, X, y) = \frac{1}{n} \sum_i \ell(y_i, f_{\hat{\theta}}(X_i))$$

التعريف في الأعلى يختصر فكرة دالة الخسارة؛ أياً كانت دالة الخسارة التي نستخدمها، خسارتنا لكامل البيانات هي متوسط الخسارة.

بالتقليل من متوسط الخسارة، نختار متغيرات النموذج التي تضبطه بناءًا على البيانات التي يتعلم منها. حتى الآن، لم نقل شيئاً عن المجتمع الإحصائي الذي أنشأ البيانات. في الحقيقة، نحن مهتمين بإجراء توقعات على جميع المجتمع الإحصائي، ليس فقط البيانات التي رأيناها.

المخاطر

إذا كانت البيانات التي رأيناها X و y تم جمعها بشكل عشوائي، فأنها تعتبر بياناتنا تعتبر متغيرات عشوائية. وإذا كانت متغيرات عشوائية، فإن متغيرات النموذج أيضاً عشوائية، في كل مرة نجمع فيها المزيد من البيانات ونضبط النموذج عليها، متغيرات النموذج $f_{\hat{\theta}}(x)$ ستتغير قليلاً.

لنفترض أننا قمنا بسحب أحد المدخلات والمخرجات z, γ من مجتمعنا الإحصائي بشكل عشوائي. الخسارة التي يكونها النموذج لهذه القيم هي:

$$\ell(\gamma, f_{\hat{\theta}}(z))$$

لاحظ أن هذه الخسارة هي متغير عشوائي؛ تتغير الخسارة عندما تستقبل قيم جديده من X و y ونقاط أخرى من z, γ من المجتمع الإحصائي.

المخاطر Risk لنموذج $f_{\hat{\theta}}$ هي النتيجة المتوقعة من الخسارة السابقة لجميع بيانات التدريب X و y وجميع النقاط z, γ في المجتمع الإحصائي:

$$R(f_{\hat{\theta}}(x)) = \mathbb{E}[\ell(\gamma, f_{\hat{\theta}}(z))]$$

لاحظ ان المخاطر هي توقع لمتغير عشوائي وليست عشوائية بحد ذاتها. القيمة المتوقعة من رمي حجر النرد ذو الست جهات بشكل عادل وعشوائي هي 3.5 على الرغم من أن عمليات الرمي نفسها عشوائية.

المثال السابق يطلق عليه أحياناً **المخاطر الحقيقة True Risk** لأنها تخبرنا عن أداء النموذج على كامل المجتمع الإحصائي. إذا تمكنا من إيجاد المخاطر الحقيقة لجميع النماذج، يمكننا ببساطة اختيار النموذج الأقل مخاطر ونكون متأكدين أن النموذج سيكون أداة أفضل من بقية النماذج على المدى البعيد على دالة الخسارة التي اخترناها.

الخطر التجريبي

الواقع ليس بهذه السهولة والروعة. إذا قمنا بتغير تعريف التوقع إلى معادة المخاطر الحقيقة، سنحصل على التالي:

$$\begin{aligned} R(f_{\hat{\theta}}) &= \mathbb{E}[\ell(\gamma, f_{\hat{\theta}}(z))] \\ &= \sum_{\gamma} \sum_{z} \ell(\gamma, f_{\hat{\theta}}(z)) P(\gamma, z) \end{aligned}$$

للتبسيط، نريد أن نعرف ما تعنيه $P(\gamma, z)$، وهي توزيع الاحتمالية العام لأي من النقاط في المجتمع الإحصائي. للأسف، إيجاد ذلك ليس سهلاً. لنفترض أننا نريد أن نتوقع قيمة الإكرامية بناءًا على عدد العملاء في الطاولة. ما هي احتمالية أن طاولة بثلاث أشخاص يقومون بإعطاء إكرامية بقيمة 14.50$؟ إذا كنا نعرف توزيع النقاط بشكل دقيقة، فلا نحتاج لجمع بيانات وضبط النموذج، سيكون لدينا معرفة بالقيمة المحتملة للإكرامية لأي عدد من العملاء في الطاولة.

على الرغم أننا لا نعرف بشكل دقيق توزيع المجتمع الإحصائي، يمكننا توقعه بناءًا على ما أطلعنا عليه في البيانات X و y. إذا قمنا بسحب قيم ل X و y بشكل عشوائي من المجتمع الإحصائي، توزيع النقاط في X و y سيكون مشابه لتوزيع المجتمع الإحصائي. لذا، نعامل X و y كمجتمعنا الإحصائي. لذا، احتمالية الظهور لأي من المدخلات والمخرجات X_i و y_i هي $\frac{1}{n}$ بما أن كل كلاهما يظهر مرة واحدة من بين كل النقاط n.

ذلك يسمح لنا بحساب **الخطر التجريبي Empirical Risk**، قيمة تقريبه للخطر الحقيقي:

$$\begin{aligned} \hat{R}(f_{\hat{\theta}}) &= \mathbb{E}[\ell(y_i, f_{\hat{\theta}}(X_i))] \\ &= \sum_{i=1}^{n} \ell(y_i, f_{\hat{\theta}}(X_i)) \frac{1}{n} \\ &= \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_{\hat{\theta}}(X_i)) \end{aligned}$$

إذا كانت البيانات لدينا ذات حجم كبير وتم سحبها بشكل عشوائي من المجتمع الإحصائي، فإن الخطر التجريبي $\hat{R}(f_{\hat{\theta}})$ سيكون قريب جداً من الخطر الحقيقي $R(f_{\hat{\theta}})$. يسمح لنا ذلك باختيار النموذج الذي يقلل من الخطر التجريبي.

لاحظ أن هذا المصطلح هو متوسط دالة الخسارة في بداية هذا الجزء! بالتقليل من متوسط الخسارة، فأننا أيضاً نقلل من الخطر التجريبي. يوضح ذلك لماذا نستخدم في العادة متوسط الخسارة كنتيجة لدالة الخسارة بدلاً مثلاً من أعلى قيمة للخسارة.

ملخص المخاطر

الخطر الحقيقي لتنبؤ النموذج يوضح لنا خسارة النموذج على المدى البعيد التي سيحصل عليها من المجتمع الإحصائي. بما أننا عادةً لا نستطيع حساب الخطر الحقيقي بشكل مباشر، فأننا نقوم بحساب الخطر التجريبي ونستخدمه لإيجاد النموذج المناسب لتوقعنا. لأن الخطر التجريبي هو متوسط الخسارة على البيانات التي رأيناها، فأننا عادةً نقلل من متوسط الخسارة عند ضبط النماذج.

انحياز وتباين النموذج

رأينا سابقاً أن لدينا مصدرين لاتخاذ قرار الأفضلية بين النماذج.

النموذج قد يكون بسيط جداً. مثلاً، نموذج خطي بسيط لن يتمكن من ضبط البيانات إذا كانت من الدرجة الثانية. يظهر هذا الخطأ بسبب **الانحياز Bias** في النموذج.

النموذج قد يقوم بضبط البيانات العشوائية الموجودة في البيانات، حتى لو قمنا بضبط بيانات من الدرجة الثانية على نموذج مماثل، سيتوقع النموذج نتائج مختلفة عن الصحيحة. هذا الخطأ يظهر في النموذج بسبب **التباين Variance**.

تحليل الانحياز والتباين

يمكننا تحليل التعريفات السابقة وفهمها بشكل أوضح باستخدام معادلة مخاطر النموذج. لنتذكر أن **المخاطر Risk** للنموذج $f_{\hat{\theta}}$ هي الخسارة المتوقعة لجميع بيانات التدريب X و y وجميع النقاط المدخلة والمخرجة z, γ في المجتمع الإحصائي:

$$R(f_{\hat{\theta}}) = \mathbb{E}[\ell(\gamma, f_{\hat{\theta}}(z))]$$

نرمز لعملية إنشاء بيانات المجتمع الإحصائي الحقيقة ب $f_\theta(x)$. القيمة الناتجة γ تنتج بواسطة بيانات المجتمع الإحصائي اضافة لقيمة عشوائية مشوشة من البيانات: $\gamma_i = f_\theta(z_i) + \epsilon$. **التشويش العشوائي Random Noise** ϵ هي قيمة عشوائية لديها متوسط يساوي صفر: $\mathbb{E}[\epsilon] = 0$.

إذا استخدمنا الخطأ التربيعي كدالة الخسارة، تصفح المعادلة السابقة كالتالي:

$$R(f_{\hat{\theta}}) = \mathbb{E}[(\gamma - f_{\hat{\theta}}(z))^2]$$

مع قليل من عمليات المعالجة الجبرية، يمكن أن نعرف المعادلة السابقة كالتالي:

$$R(f_{\hat{\theta}}) = (\mathbb{E}[f_{\hat{\theta}}(z)] - f_\theta(z))^2 + \mathrm{Var}(f_{\hat{\theta}}(z)) + \mathrm{Var}(\epsilon)$$

المصطلح الأول في المعادلة، $(\mathbb{E}[f_{\hat{\theta}}(z)] - f_\theta(z))^2$، هي التعبير الرياضي للانحياز في النموذج. (يمكننا القول تقنياً إن المصطلح هو تربيع الانحياز bias^2). الانحياز سيساوي صفر على المدى البعيد إذا كان اختيارنا للنموذج $f_{\hat{\theta}}(z)$ يتوقع نفس النتائج التي تم نتجت من خطوات معالجة المجتمع الإحصائي $f_\theta(z)$. يكون الانحياز عالي إذا كان النموذج الذي اخترناه يتوقع نتائج خاطئة في خطوات معالجة المجتمع الإحصائي حتى لو كانت كامل المجتمع الإحصائي كبيانات تدريب.

المصطلح الثاني في المعادلة، $\text{Var}(f_{\hat{\theta}}(z))$، هو تباين النموذج. يكون التباين أقل عندما تكون توقعات النموذج لا تتغير كثيراً عند تدريبه على بيانات مختلفة من المجتمع الإحصائي. يكون التباين عالي عندما يكون التغير كبير عند تدريبه على بيانات مختلفة من المجتمع الإحصائي.

المصطلح الأخير في المعادلة، $\text{Var}(\epsilon)$، يعني الخطأ الغير قابل للإختزال أو التشويش الناتج عند بناء البيانات أو جمعها. يكون أقل عندما يكون بناء البيانات وجمعها دقيق. والعكس يكون أعلى عندما يكون التشويش عالي في البيانات.

مشتقة تحليل الانحياز والتباين

أولاً، نبدأ بمتوسط الخطأ التربيعي:

$$\mathbb{E}[(\gamma - f_{\hat{\theta}}(z))^2]$$

ثم نوسع التربيع ونطبق خطية التوقع Linearity of Expectation:

$$= \mathbb{E}[\gamma^2 - 2\gamma f_{\hat{\theta}} + f_{\hat{\theta}}(z)^2$$

$$= \mathbb{E}[\gamma^2] - \mathbb{E}[2\gamma f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

لأن γ و $f_{\hat{\theta}}(z)$ مستقلان (نتائج النموذج وما اطلع عليه من المجتمع الإحصائي لا يعتمدان على بعضهما)، يمكننا القول إن $\mathbb{E}[2\gamma f_{\hat{\theta}}(z)] = \mathbb{E}[2\gamma]\mathbb{E}[f_{\hat{\theta}}(z)]$. ثم نقوم بتعويض $f_\theta(z) + \epsilon$ بدلاً من γ:

$$= \mathbb{E}[(f_\theta(z) + \epsilon)^2] - \mathbb{E}[2(f_\theta(z) + \epsilon)]\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

للتبسيط أكثر: (لاحظ أن $\mathbb{E}[f_\theta(z)] = f_\theta(z)$ لأن $f_\theta(z)$ دالة حتمية، إذا أعطيت نقطة معينة z.)

$$= \mathbb{E}[f_\theta(z)^2 + 2f_\theta(z)\epsilon + \epsilon^2] - (2f_\theta(z) + \mathbb{E}[2\epsilon])\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

تطبيق خطية التوقع مرة أخرى:

$$= f_\theta(z)^2 + 2f_\theta(z)\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2] - (2f_\theta(z) + 2\mathbb{E}[\epsilon])\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

لاحظ أن $\big(\mathbb{E}[\epsilon] = 0\big) => \big(\mathbb{E}[\epsilon^2] = \text{Var}(\epsilon)\big)$ لأن $\text{Var}(\epsilon) = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2$:

$$= f_\theta(z)^2 + \text{Var}(\epsilon) - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

يمكننا الآن إعادة كتابة المعادلة كالتالي:

$$= f_\theta(z)^2 + \text{Var}(\epsilon) - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2] - \mathbb{E}[f_{\hat{\theta}}(z)]^2 + \mathbb{E}[f_{\hat{\theta}}(z)]^2$$

لأن $\mathbb{E}[f_{\hat{\theta}}(z)^2] - \mathbb{E}[f_{\hat{\theta}}(z)]^2 = Var(f_{\hat{\theta}}(z))$:

$$= f_\theta(z)^2 - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)]^2 + Var(f_{\hat{\theta}}(z)) + \text{Var}(\epsilon)$$

$$= (f_\theta(z) - \mathbb{E}[f_{\hat{\theta}}(z)])^2 + Var(f_{\hat{\theta}}(z)) + \text{Var}(\epsilon)$$

$$= \text{bias}^2 + \text{model variance} + \text{noise}$$

لاختيار نموذج يؤدي بشكل ممتاز، نسعى دائماً أن نحصل على مخاطر أقل. ولتقليلها، نحاول تقليل الانحياز، التباين والتشويش. تقليل التشويش يتطلب عادةً تحسين في طريقة جمع البيانات. للتقليل من الانحياز والتباين، يجب علينا ضبط النماذج وتعقيدها. النماذج البسيطة يكون فيها الانحياز مرتفع؛ النماذج المعقدة جداً لديها تباين عالي. هذا هو مفهوم *المقايضة بين الانحياز والتباين bias-variance tradeoff*، مشكلة أساسية نواجهها دائماً عند اختيار نماذج التنبؤ.

مثال: الإنحدار الخطي وموجات الجيب Sine Waves

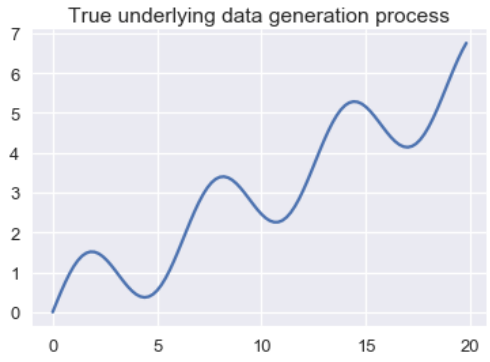
لنفترض أننا نحاول إنشاء نموذج للدالة المتأرجحة التالية:

```
from collections import namedtuple
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

np.random.seed(42)

Line = namedtuple('Line', ['x_start', 'x_end', 'y_start', 'y_end'])

def f(x): return np.sin(x) + 0.3 * x
```

```
def noise(n):
    return np.random.normal(scale=0.1, size=n)

def draw(n):
    points = np.random.choice(np.arange(0, 20, 0.2), size=n)
    return points, f(points) + noise(n)

def fit_line(x, y, x_start=0, x_end=20):
    clf = LinearRegression().fit(x.reshape(-1, 1), y)
    return Line(x_start, x_end, clf.predict([[x_start]])[0], clf.predict([[x_end]])[0])

population_x = np.arange(0, 20, 0.2)
population_y = f(population_x)

avg_line = fit_line(population_x, population_y)


datasets = [draw(100) for _ in range(20)]
random_lines = [fit_line(x, y) for x, y in datasets]
```

قام الكاتب في الكود البرمجي السابق إنشاء بيانات عشوائية لغرض العمل عليها في هذا الجزء من الدرس، الناتج من هذا الكود يظهر في الرسم البياني التالي

```
plt.plot(population_x, population_y)
plt.title('True underlying data generation process');
```

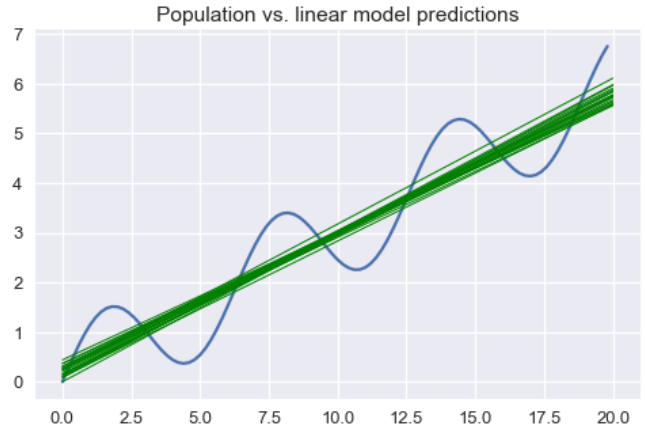


إذا حاولنا أخذ مجموعه بيانات من المجتمع الإحصائي في هذه الرسم، قد نصل إلى الشكل التالي:

```
xs, ys = draw(100)
plt.scatter(xs, ys, s=10)
plt.title('One set of observed data');
```

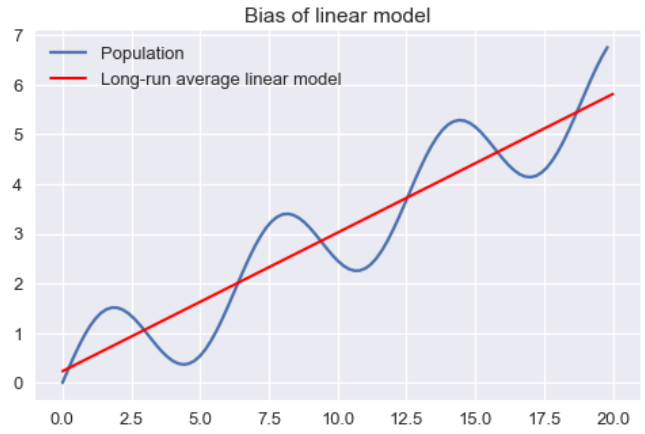
لنفترض أننا قمنا بسحب مجموعات مختلفة من البيانات من هذا المجتمع الإحصائي وقمنا بضبط نموذج خطي بسيط لكل مجموعة. في الأسفل، قمنا برسم شكل خطي لبيانات المجتمع الإحصائي الناتجة باللون الأزرق و توقعات النموذج باللون الأخضر:

```
plt.figure(figsize=(8, 5))
plt.plot(population_x, population_y)

for x_start, x_end, y_start, y_end in random_lines:
    plt.plot([x_start, x_end], [y_start, y_end], linewidth=1, c='g')

plt.title('Population vs. linear model predictions');
```



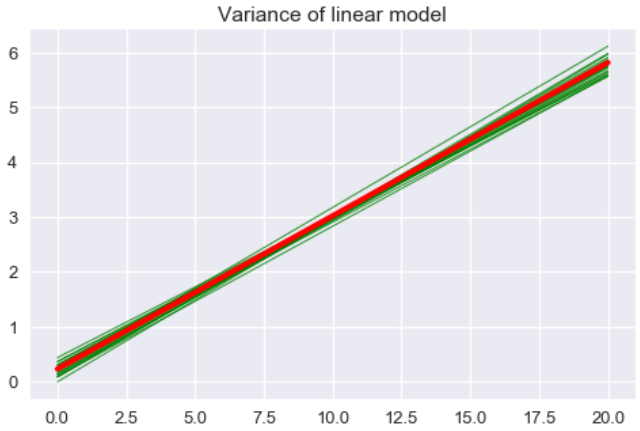
الرسم البياني السابق يوضح أن توقع النموذج الخطي سينتج أخطاء. يمكننا حل هذه الأخطاء باستخدام الانحياز، التباين والتشويش الغير قابل للإختزال. نوضح الانحياز في نموذجنا عن طريق إظهار أن متوسط النموذج الخطي على المدى البعيد سيتوقع نتائج مختلفة عن تلك في المجتمع الإحصائي:

```
plt.figure(figsize=(8, 5))
xs = np.arange(0, 20, 0.2)
plt.plot(population_x, population_y, label='Population')

plt.plot([avg_line.x_start, avg_line.x_end],
         [avg_line.y_start, avg_line.y_end],
         linewidth=2, c='r',
         label='Long-run average linear model')
plt.title('Bias of linear model')
plt.legend();
```



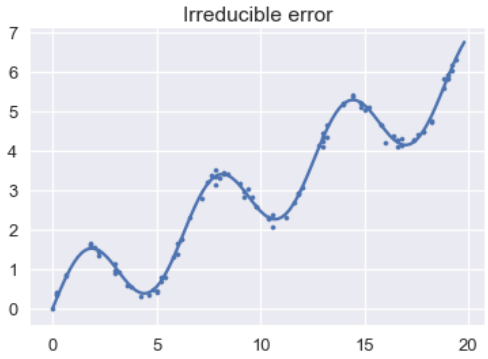
التباين للنموذج هو مدى اختلاف توقعات النموذج حول متوسط النموذج على المدى البعيد:

```
plt.figure(figsize=(8, 5))
for x_start, x_end, y_start, y_end in random_lines:
    plt.plot([x_start, x_end], [y_start, y_end], linewidth=1, c='g', alpha=0.8)

plt.plot([avg_line.x_start, avg_line.x_end],
         [avg_line.y_start, avg_line.y_end],
         linewidth=4, c='r')

plt.title('Variance of linear model');
```



أخيراً، نستعرض الخطأ الغير قابل للإختزال عن طريق إظهار انحراف النقاط في بيانات المجتمع الإحصائي:

```
plt.plot(population_x, population_y)


xs, ys = draw(100)
plt.scatter(xs, ys, s=10)
plt.title('Irreducible error');
```



التطبيق العملي للانحياز والتباين

في عالمٍ مثالي، نقوم بالتقليل من الخطأ المتوقع من تنبؤ النموذج لجميع النقاط الداخلة والخارجة في المجتمع الإحصائي. ولكن، عملياً، نحن لا نعرف خطوات بناء البيانات ولذلك لن نستطيع بشكل دقيق تحديد انحياز، تباين والخطأ الغير قابل للإختزال للنموذج. بدلاً من ذلك، نستخدم البيانات التي مرت علينا وقيمه قريبه لما في المجتمع الإحصائي.

كما رأينا في الأعلى، الوصول إلى قيمة أقل للخطأ في بيانات التدريب لا يعني أن النموذج سيحصل على قيمه أقل أيضاً في بيانات الاختبار. من السهل الحصول على نموذج بقيمة انحياز قليله جداً وبالتالي قيمة خطأ أقل في بيانات التدريب عن طريق ضبط النموذج باستخدام منحنى يمر على جميع بيانات التدريب. ولكن هذا النموذج سيكون تباينه عالي جداً والذي يؤدي إلى خطأ عالي في بيانات الاختبار. على العكس، النموذج الذي يتوقع قيمه ثابتة لدية تباين أقل و انحياز عالي. أساساً، يحدث ذلك بسبب أن خطأ التدريب يعكس انحياز النموذج ولكن ليس تباينه؛ ولكن خطأ الاختبار يعكس كليهما. للتقليل من خطأ الاختبار، يحتاج نموذجنا أن يحقق انحياز وتباين قليلان معاً. للقيام بذلك، نحتاج لطريقة لمحاكاة خطأ الاختبار دون استخدام بيانات الاختبار. يتم ذلك عادةً باستخدام التحقق المتقاطع Cross-Validation.

النقاط الأساسية

المقايضة بين الانحياز والتباين تسمح لنا بشكل دقيق وصف ظواهر النماذج التي رأيناها سابقاً.

يحدث فرط التعميم Underfitting عادةً بسبب الانحياز؛ فرط التخصيص يحدث بسبب التباين.

جمع المزيد من البيانات يقلل من التباين. مثلاً، تباين نموذج الانحدار الخطي يقل بمعدل عامل $1/n$، هنا n هي عدد النقاط في البيانات. لذا، مضاعفة حجم البيانات يقلل التباين إلى النصف، وسحب المزيد من البيانات سيقلل التباين إلى صفر. إحدى الخطوات الشائعة هي اختيار نموذج بانحياز قليل وتباين عالي (مثلاً شبكة عصبيه) ومن ثم سحب المزيد من البيانات للتقليل من تباين النموذج إلى ان يصل لقيمه قليله تمكنه من القيام بتوقعات صحيحه. على الرغم فاعليتها عملياً، إلى أن جمع المزيد من البيانات لهذه النماذج عادةً ما يأخذ وقت و جهد ومصاريف أكثر.

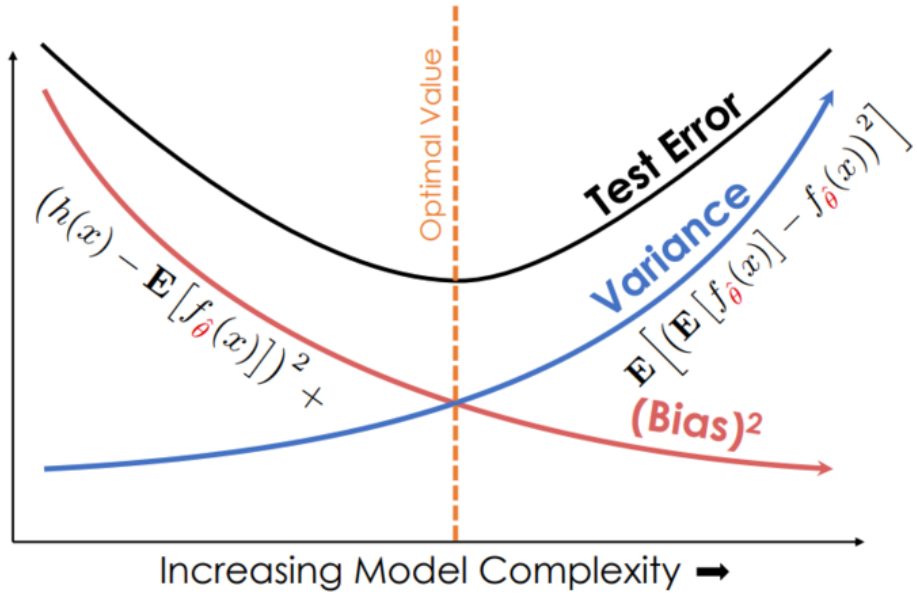
جمع المزيد من البيانات يقلل الانحياز إذا تم ضبط النموذج على بيانات المجتمع الإحصائي كاملة. إذا لم نتمكن من ضبط النموذج على كامل المجتمع الإحصائي (كما في المثال السابق)، حتى لو كان لدينا عدد لا نهائي من البيانات لن نتمكن من التخلص من الانحياز.

إضافة خصائص Features مفيدة للبيانات، مثلاً التربيع عندما تكون البيانات الأساسية من الدرجة الثانية، تقلل من الانحياز. إضافة خصائص غير مفيدة نادراً منا تزيد من الانحياز.

إضافة خصائص سواء كانت مفيدة أم لا، عادةً ما تزيد من التباين كون كل خاصية نضيفها تزيد من المتغيرات في النموذج. بشكل عام، النماذج ذات المتغيرات الكثيرة لديها أشكال مختلفة من المتغيرات ولذلك لديها تباين أعلى من نماذج بمتغيرات أقل. للزيادة من دقة توقعات النموذج، إضافة خاصية جديده يجب أن تقلل من الانحياز أكثر مما تزيد من التباين.

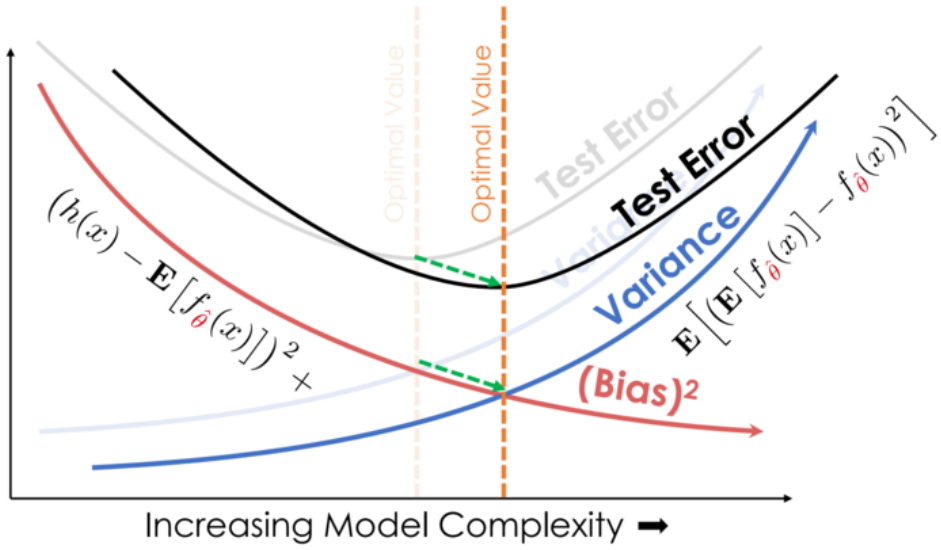
حذف الخصائص عادةً ما يزيد من الانحياز وقد يتسبب بفرط التعميم. مثلاً، نموذج خطي بسيط لديه انحياز أعلى من نفس النموذج إذا أضيفت له خاصية تربيعية. إذا تم إنشاء البيانات باستخدام التربيع، فإن النموذج الخطي البسيط سيتناسب مع البيانات.

في الرسم البياني التالي، المحور السيني يقيس تعقيد النموذج والمحور الصادي يقيس حجمه. لاحظ أنه كلما زاد تعقيد النموذج، يقل الانحياز بشكل واضح وبينما يزيد التباين بنفس الشكل. عندما نختار نموذجاً معقد، خطأ الاختبار في البداية يقل ثم يزيد حتى يصل تباين النموذج ويتجاوز الانحياز أثناء هبوطه:



كما هو واضح في الرسم، النموذج ذو التعقيد العالي قد يصل إلى خطأ قليل في بيانات التدريب ولكن قد يفشل في تعميم هذه النتائج على بيانات الاختبار بسبب تباينه العالي. بشكل آخر، النموذج بتعقيد أقل سيكون ذو تباين أقل ولكن قد يفشل أيضاً في التعميم بسبب انحيازه العالي. لاختيار النموذج المفيد، يجب أن نصل إلى التوازن بين انحياز وتباين النموذج.

كلما أضفنا المزيد إلى البيانات، نقوم بتحريك المنحنى في رسمنا البياني إلى اليمين والأسفل، نقلل من الانحياز والتباين:



ملخص انحياز وتباين النموذج

مقايضة الانحياز والتبيان تكشف لنا عن مشكلة أساسية في النمذجة. من أجل التقليل من مخاطر النموذج، فعلينا أن نستخدم مزيجاً من هندسة الخصائص، اختيار النماذج والتحقق المتقاطع للوصول إلى توازن بين الانحياز والتباين.

التحقق المتقاطع

في الجزء السابق، رأينا أننا نحتاج لطريقة أدق لمحاكاة خطأ الاختبار للتحكم بالمقايضة بين الانحياز والتباين. للتأكيد، خطأ التدريب قليل جداً، لأننا نضبط النموذج على بيانات التدريب. نريد اختبار نموذج دون الحاجة لاستخدام بيانات الاختبار، لذا نقسم بيانات التدريب مرة أخرى إلى بيانات تحقق. يسمح لنا **التحقق المتقاطع Cross-validation** بتوقع خطأ النموذج باستخدام بيانات يطلع عليها مرة واحدة عن طريق فصل بين البيانات المستخدمة في التدريب والبيانات المستخدمة لاختيار النموذج ودقته.

تقسيم بيانات التدريب والتحقق والاختبار

إحدى الطرق لتطبيق ذلك على البيانات هي تقسيمها إلى ثلاث أقسام:

- بيانات التدريب: البيانات التي ستستخدم لضبط النموذج.
- بيانات التحقق: البيانات التي ستستخدم لاختيار الخصائص.
- بيانات الاختبار: البيانات التي ستقرر النتيجة النهائية لدقة النموذج.

بعد التقسيم، نختار الخصائص والنموذج بناءًا على:

- لكل خاصية محتملة في البيانات، نقوم بضبطها على النموذج باستخدام بيانات التدريب. الخطأ في هذا النموذج هو *خطأ التدريب Training Error*.
- تحقق من كل نتيجة خطأ لكل نموذج في بيانات التحقق: يطلق على ذلك *خطأ التحقق Validation Error*. قم باختيار النموذج صاحب أقل خطأ تحقق. سيكون ذلك هو الخيار النهائي للنموذج وخصائصه.
- قم بحساب *خطأ الاختبار Test Error*، الخطأ في النموذج النهائي على بيانات الاختبار. هذه هي الدقة النهائية للنموذج. محظور لنا من التعديل الخصائص أو النموذج للتقليل من خطأ الاختبار؛ فعل ذلك سيحول بيانات الاختبار إلى بيانات تحقق. بدلاً من ذلك، يجب علينا جمع بيانات جديده للاختبار بعد القيام بالتعديلات على الخصائص أو النموذج.

تسمح لنا هذه الخطوات بشكل دقيق من تحديد النموذج الذي نريد استخدامه بدلاً من استخدام خطأ التدريب وحده. باستخدام التحقق المتقاطع، يمكننا اختبار النموذج على بيانات لم يتم ضبطه عليها، محاكاة خطأ الاختبار دون استخدام بيانات الاختبار. يسمح لنا ذلك بمعرفة قدرة النموذج على التعامل مع بيانات لم يرها من قبل.

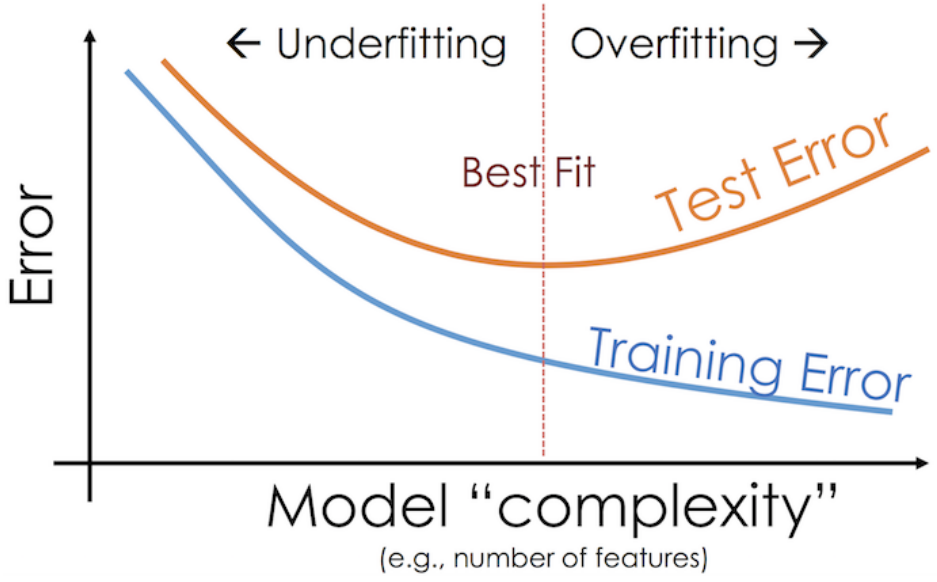
حجم تقسيم بيانات التدريب والتحقق والاختبار

تقسيم بيانات التدريب والتحقق والاختبار عادة ما يكون فيه 70% بيانات تدريب، 15% بيانات تحقق وال 15% الباقية بيانات اختبار. الزيادة في عدد بيانات التدريب يساعد على رفع دقة النموذج ولكنه يسبب مزيداً من الاختلاف في بيانات التحقق والاختبار. ذلك بسبب أن بيانات التحقق والاختبار ذات الحجم الصغير غير كافيه لتمثيل البيانات.

خطأ التدريب و خطأ الاختبار

النموذج يكون غير مفيد لنا إذا فشل في التعامل مع بيانات من المجتمع الإحصائي لم يرها من قبل. خطأ الاختبار يقدم لنا أدق تمثيل لأداء النموذج على بيانات جديده كوننا لا نستخدم بيانات الاختبار في تدريب النموذج أو اختيار الخصائص.

بشكل عام، يقل خطأ التدريب كلما أضفنا تعقيداً أكثر للنموذج بإضافة المزيد من الخصائص أو خطوات معقدة في التوقع. خطأ الاختبار، يقل حتى نقطة معينة ثم يزيد إلى أن يتم ضبطه على بيانات التدريب. ذلك بسبب أنه في البداية، الانحياز يقل أكثر من زيادة التباين. في النهاية، تتجاوز الزيادة في التباين النزول في الانحياز.

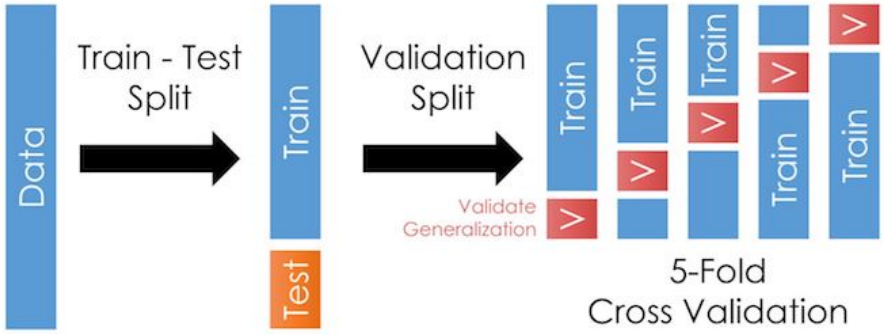


التحقق المتقاطع K-Flod

طريقة تقسيم بيانات التدريب والتحقق والاختبار جيدة لمحاكاة نتيجة خطأ الاختبار باستخدام بيانات التحقق. ولكن، عند التقسيم سيكون حجم البيانات أقل في التدريب. أيضاً، بهذه الطريقة قد يكون خطأ التحقق يميل إلى الزيادة في التباين بسبب أن تقييم الخطأ قد يعتمد بشكل كبير على مكان تواجد القيم في بيانات التدريب أو التحقق.

لمواجهة هذه المشكلة، يمكننا تطبيق تقيم التدريب والتحقق أكثر من مرة على نفس البيانات. يتم تقسيم البيانات إلى تقسيمات بعدد k متساوية في الحجم (k تقسيمات) ، ويتم اعادة تقسيم التدريب والتحقق بعدد k مرات. في كل مرة، إحدى التقسيمات في k تستخدم كبيانات تحقق، والقيم المتبقية $k-1$ تستخدم كبيانات تدريب. نقوم بتحديد النتيجة النهائية لخطأ التحقق للنموذج بإيجاد متوسط خطأ التحقق في كل تقسيم. يطلق على هذه الطريقة بـ **K-Flod التحقق المتقاطع K-Flod cross-validation**.

الرسم التالي يشرح الطريقة عند استخدامها التقسيم 5 مرات:



ما يميز هذه الطريقة أن كل نقطة في البيانات تستخدم مرة واحد فقط في بيانات التحقق و في بيانات التدريب $k-1$ مرات. عادةً، تكون k بين 5 و 10، ولكن تبقى قيمة k غير ثابتة. عندما تكون k رقماً صغيراً، الخطأ يكون تباينه قليلاً (الكثير من بيانات التحقق) ولكن لدية انحياز عالي (القليل من بيانات التدريب). على العكس، عندما تكون قيمة k عالية قيمة الخطأ تكون أقل تحيزاً ولكن تباينها عالي.

التحقق المتقاطع k-flod يأخذ وقتاً أطول من تقسيم بيانات التدريب والتحقق في عملية الحساب لأننا عادةً نحتاج لإعادة ضبط كل نموذج مع كل عملية تقسيم. ولكن، عملية الحساب تعطي نتائج أكثر دقة لخطأ التحقق عن طريق حساب متوسط أكثر من قيمة خطأ لكل نموذج.

مكتبة `scikit-learn` توفر طريقة جاهزة `sklearn.model_selection.KFold` لتطبيق التحقق المتقاطع k-flod.

مقايضة الانحياز والتباين

يساعدنا التحقق المتقاطع في التحكم بمقايضة الانحياز والتباين بشكل أكثر دقة. حدسياً، خطأ التحقق يقيم خطأ الاختبار عن طريق التحقق من أداء النموذج على بيانات لم تستخدم في التدريب؛ يسمح لنا ذلك بتوقع انحياز وتباين النموذج. التحقق المتقاطع K-flod يظهر أيضاً حقيقة أن التشويش في بيانات الاختبار فقط تأثر على التشويش في مصطلح الانحياز والتباين بينما التشويش في بيانات التدريب يؤثر على الانحياز وتباين النموذج. لاختيار النموذج النهائي للبدء باستخدامه، نختار النموذج ذو قيمة خطأ تحقق أقل.

مثال: اختيار النموذج لبيانات تقييم الآيس كريم

سنتعرف على عملية اختيار النموذج الكاملة، بالإضافة للتحقق المتقاطع، لاختيار نموذج يتوقع تقييم الآيس كريم بناءًا على حلاوة نكهة الآيس كريم. جميع بيانات الآيس كريم إضافة لمخطط التشتت والتقييم العام بالإضافة إلى حلاوة نكهة الآيس كريم هي كالتالي:

لتحميل البيانات icecream.csv اضغط هنا.

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

ice = pd.read_csv('icecream.csv')
transformer = PolynomialFeatures(degree=2)
X = transformer.fit_transform(ice[['sweetness']])

clf = LinearRegression(fit_intercept=False).fit(X, ice[['overall']])
xs = np.linspace(3.5, 12.5, 300).reshape(-1, 1)
rating_pred = clf.predict(transformer.transform(xs))

temp = pd.DataFrame(xs, columns = ['sweetness'])
temp['overall'] = rating_pred

np.random.seed(42)
x_devs = np.random.normal(scale=0.2, size=len(temp))
y_devs = np.random.normal(scale=0.2, size=len(temp))
temp['sweetness'] = np.round(temp['sweetness'] + x_devs, decimals=2)
temp['overall'] = np.round(temp['overall'] + y_devs, decimals=2)

ice = pd.concat([temp, ice])

ice
```

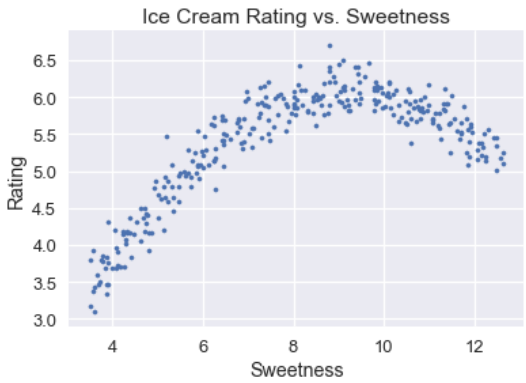
	sweetness	overall
0	3.6	3.09

	sweetness	overall
1	3.5	3.17
2	3.69	3.46
...	...	...
6	11	5.9
7	11.7	5.5
8	11.9	5.4

```
309 rows × 2 columns
```

في الكود البرمجي السابق قام الكاتب بتوليد بيانات عشوائية مقاربة للبيانات الأصلية في ملف ice.csv

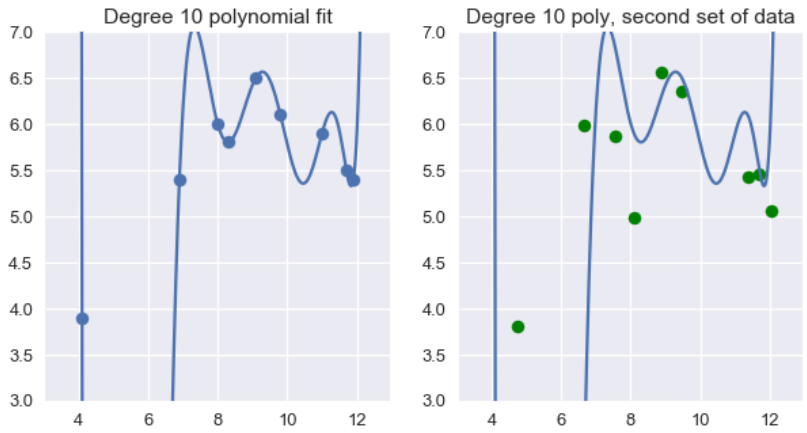
```
plt.scatter(ice['sweetness'], ice['overall'], s=10)
plt.title('Ice Cream Rating vs. Sweetness')
plt.xlabel('Sweetness')
plt.ylabel('Rating');
```



باستخدام خصائص متعددة الحدود من الدرجة العاشرة على 9 مدخلات في ملف البيانات تكون لنا نموذج مثالي لتلك النقاط. للأسف، سيفشل هذا النموذج في التعميم لبيانات لم يراها من قبل في المجتمع الإحصائي:

```
ice2 = pd.read_csv('icecream.csv')
trans_ten = PolynomialFeatures(degree=10)
X_ten = trans_ten.fit_transform(ice2[['sweetness']])
y = ice2['overall']

clf_ten = LinearRegression(fit_intercept=False).fit(X_ten, y)
```

```
np.random.seed(1)
x_devs = np.random.normal(scale=0.4, size=len(ice2))
y_devs = np.random.normal(scale=0.4, size=len(ice2))

plt.figure(figsize=(10, 5))

plt.subplot(121)
plt.scatter(ice2['sweetness'], ice2['overall'])
xs = np.linspace(3.5, 12.5, 1000).reshape(-1, 1)
ys = clf_ten.predict(trans_ten.transform(xs))
plt.plot(xs, ys)
plt.title('Degree 10 polynomial fit')
plt.ylim(3, 7);

plt.subplot(122)
ys = clf_ten.predict(trans_ten.transform(xs))
plt.plot(xs, ys)
plt.scatter(ice2['sweetness'] + x_devs,
            ice2['overall'] + y_devs,
            c='g')
plt.title('Degree 10 poly, second set of data')
plt.ylim(3, 7);
```



بدلاً من الطريقة السابقة، أولاً نقسم البيانات إلى تدريب، تحقق واختبار باستخدام الدالة `sklearn.model_selection.train_test_split` من مكتبة `scikit-learn` للقيام بفصل يساوي 70/30% تدريب-اختبار.

```
from sklearn.model_selection import train_test_split

test_size = 92

X_train, X_test, y_train, y_test = train_test_split(
    ice[['sweetness']], ice['overall'], test_size=test_size, random_state=0)


print(f'  Training set size: {len(X_train)}')
print(f'      Test set size: {len(X_test)}')
```

```
  Training set size: 217
      Test set size: 92
```

نقوم الآن بضبط نماذج الانحدار متعددة الحدود باستخدام بيانات التدريب، واحد لكل درجة في متعددة الحدود من 1 إلى 10.

```
# أولاً، نظيف خصائص متعددة الحدود إلى X_train
transformers = [PolynomialFeatures(degree=deg)
                for deg in range(1, 11)]
X_train_polys = [transformer.fit_transform(X_train)
                 for transformer in transformers]

# عرض X_train
# مع متجهة خصائص من الدرجة الخامسة
X_train_polys[4]
```

```
array([[     1.  ,      8.8 ,     77.44,    681.47,   5996.95,  52773.19],
       [     1.  ,     10.74,    115.35,   1238.83,  13305.07, 142896.44],
       [     1.  ,      9.98,     99.6 ,    994.01,   9920.24,  99003.99],
       ...,
       [     1.  ,      6.79,     46.1 ,    313.05,   2125.59,  14432.74],
       [     1.  ,      5.13,     26.32,    135.01,    692.58,   3552.93],
       [     1.  ,      8.66,     75.  ,    649.46,   5624.34,  48706.78]])
```

ثم نقوم بتطبيق التحقق المتقاطع ب 5 تقسيمات على البيانات العشرة. للقيام بذلك، سنقوم بتعريف دالة تقوم بالتالي:

- استخدام `KFold.split` للحصول على 5 تقسيمات في بيانات التدريب. لاحظ أن `split` تعود لنا بمؤشرات `indices` للبيانات في ذلك الفصل.
- لكل عملية فصل، استخرج لنا الأسطر والأعمدة بناءاً على مؤشرات الفصل `indices` والخصائص.
- اضبط النموذج الخطي على بيانات التدريب المقسمة.
- احسب متوسط الخطأ التربيعي في بيانات التحقق المقسمة.
- أوجد لنا متوسط الخطأ لكل تقسيمات التحقق المتقاطع.

```
from sklearn.model_selection import KFold

def mse_cost(y_pred, y_actual):
    return np.mean((y_pred - y_actual) ** 2)

def compute_CV_error(model, X_train, Y_train):
    kf = KFold(n_splits=5)
    validation_errors = []
```

```
    for train_idx, valid_idx in kf.split(X_train):
        # تقسيم البيانات
        split_X_train, split_X_valid = X_train[train_idx], X_train[valid_idx]
        split_Y_train, split_Y_valid = Y_train.iloc[train_idx], Y_train.iloc[valid_idx]

        # ضبط النموذج على تقسيمة التدريب
        model.fit(split_X_train,split_Y_train)

        # حساب متوسط الخطأ التربيعي على بيانات التحقق
        error = mse_cost(split_Y_valid,model.predict(split_X_valid))

        validation_errors.append(error)


    # متوسط خطأ بيانات التحقق جميعها
    return np.mean(validation_errors)
```

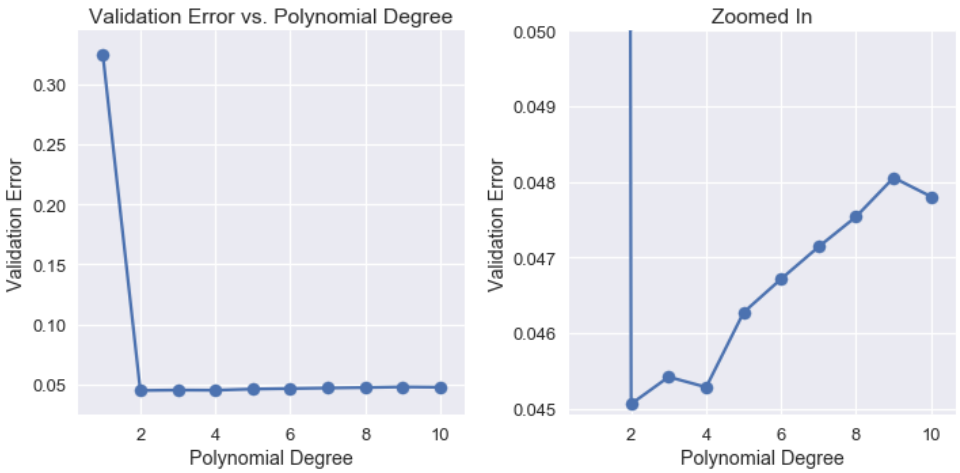
```
# قمنا بتدريب نموذج إنحدار خطي لكل البيانات وطبقنا التحقق المتقاطع
# حددنا fit_intercept=False
# في نموذج الإنحدار الخطي لأن محول PolynomialFeatures
# يقوم بإضافة عمود الانحياز بدلاً عنا.

cross_validation_errors = [compute_CV_error(LinearRegression(fit_intercept=False), X_train_poly, y_train)
                    for X_train_poly in X_train_polys]
```

```
cv_df = pd.DataFrame({'Validation Error': cross_validation_errors}, index=range(1, 11))
cv_df.index.name = 'Degree'
pd.options.display.max_rows = 20
display(cv_df)
pd.options.display.max_rows = 7
```

	Validation Error
Degree	
1	0.32482
2	0.04506
3	0.045418
4	0.045282
5	0.046272
6	0.046715
7	0.04714
8	0.04754
9	0.048055
10	0.047805

يمكن أن نلاحظ أنه عندما نرفع قيمة درجة متعددة الحدود، تقل قيمة خطأ التحقق وتزيد مرة أخرى:

```
plt.figure(figsize=(10, 5))

plt.subplot(121)
plt.plot(cv_df.index, cv_df['Validation Error'])
plt.scatter(cv_df.index, cv_df['Validation Error'])
plt.title('Validation Error vs. Polynomial Degree')
plt.xlabel('Polynomial Degree')
plt.ylabel('Validation Error');

plt.subplot(122)
plt.plot(cv_df.index, cv_df['Validation Error'])
plt.scatter(cv_df.index, cv_df['Validation Error'])
plt.ylim(0.044925, 0.05)
plt.title('Zoomed In')
plt.xlabel('Polynomial Degree')
plt.ylabel('Validation Error')

plt.tight_layout();
```



عند مراجعة نتائج خطأ التحقق يظهر لنا أن النموذج المثالي أستخدم خصائص متعددة الحدود من الدرجة الثانية. لذا، نختار نموذج متعددة الحدود من الدرجة الثانية كنموذجنا النهائي وضبطه على جميع بيانات التدريب. ثم نقوم بحساب خطأ النموذج على بيانات الاختبار:

```
best_trans = transformers[1]
best_model = LinearRegression(fit_intercept=False).fit(X_train_polys[1], y_train)

training_error = mse_cost(best_model.predict(X_train_polys[1]), y_train)
validation_error = cross_validation_errors[1]
test_error = mse_cost(best_model.predict(best_trans.transform(X_test)), y_test)

print('Degree 2 polynomial')
print(f'  Training error: {training_error:0.5f}')
print(f'Validation error: {validation_error:0.5f}')
print(f'      Test error: {test_error:0.5f}')
```

```
Degree 2 polynomial
  Training error: 0.04409
Validation error: 0.04506
      Test error: 0.04698
```

للتطبيق مستقبلاً، توفر مكتبة `scikit-learn` طريقة `cross_val_predict` للقيام أوتوماتيكياً بالتحقق المتقاطع، لذا لا نحتاج أن نقوم بأنفسنا بفصل بيانات التدريب والتحقق.

أيضاً، لاحظ أن خطأ الاختبار أعلى من خطأ التحقق والذي أيضاً أعلى من خطأ التدريب. خطأ التدريب يجب أن يكون الأقل لأن النموذج تم ضبطه على بيانات التدريب. ضبط النموذج يقلل من متوسط الخطأ التدريبي للبيانات. خطأ التحقق والاختبار عادةً ما تكون أعلى من خطأ التدريب لأن حساب الخطأ فيها تم على بيانات غير معروفة ولم يسبق أن رآها في النموذج.

ملخص التحقق المتقاطع

استخدمنا الوسيلة المفيدة دائماً التحقق المتقاطع للتحكم بمقايضة الانحياز والتباين. بعد حساب فصل بيانات التدريب، التحقق والاختبار على البيانات الأصلية، نستخدم الخطوات التالية لتدريب واختيار النموذج:

- لكل مجموعة من الخصائص، نقوم بضبط بيانات التدريب عليها. خطأ النموذج على بيانات التدريب هو *خطأ التدريب Training Error*.
- تحقق من الخطأ لكل نموذج في بيانات التحقق باستخدام التحقق المتقاطع k-flod cross validation: الخطأ هنا هو *خطأ التحقق Validation Error*. نختار النموذج الذي حقق أقل قيمة في خطأ التحقق. هنا نكون قد اخترنا خياراتنا النهائية لخصائص النموذج.
- نحسب *خطأ الاختبار Test Error*، خطأ النموذج النهائي على بيانات الاختبار. هذه آخر خطوه لمعرفة دقة النموذج. يجب ألا نقوم بأي تعديل في النموذج لزيادة خطأ الاختبار؛ القيام بذلك يحول بيانات الاختبار إلى بيانات تحقق. عند القيام بذلك، نحتاج لجمع بيانات اختبار جديده بعد القيام بأي تعديلات في النموذج.

الضبط

مقدمة

يمكن لهندسة الخصائص أن تقدم لنا معلومات مهمة عن خطوات توليد البيانات في النموذج. ولكن، إضافة الخصائص إلى البيانات عادة ما يزيد من التباين في نموذجنا ويمكن لذاك أن يسيء من أداءه بشكل عام. بدلاً من إنشاء الخصائص بشكل عشوائي، يمكننا استخدام طريقة تسمى بـ **الضبط Regularization** للتقليل من التباين في نموذجنا مع الاستمرار بتقديم أكبر قدر من المعلومات عن البيانات.

فكرة الضبط

لنبدأ النقاش عن الضبط باستخدام المثال التالي لتوضيح أهميته.

بيانات سد المياه

البيانات التالية توضح كمية المياه باللتر التي تتدفق من سد المياه في اليوم و قيمة التغير في مستوى المياه في نفس اليوم بالمتر.

لتحميل البيانات water_large.csv اضغط هنا.

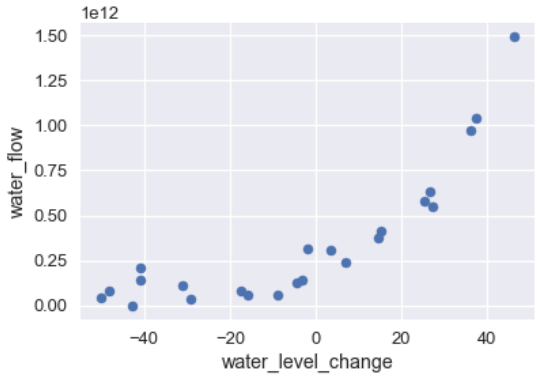
```
df = pd.read_csv('water_large.csv')
df
```

	water_level_change	water_flow
0	15.936758-	6.04E+10
1	29.152979-	3.32E+10
2	36.189549	9.73E+11
...	...	...
20	7.08548	2.36E+11
21	46.282369	1.49E+12
22	14.612289	3.78E+11

```
23 rows × 2 columns
```

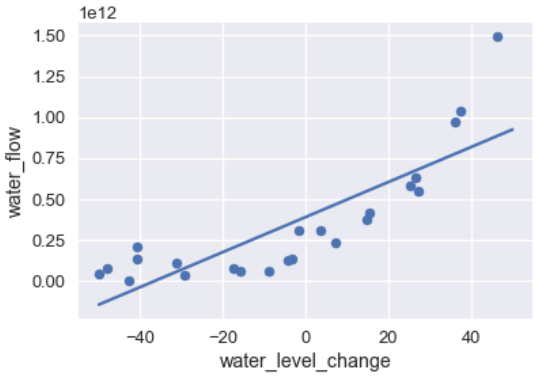
عند رسم البيانات نلاحظ الاتجاه التصاعدي لتدفق المياه كلما كانت مستويات المياه إيجابية:

```
df.plot.scatter(0, 1, s=50);
```



لنمذجة هذا النمط، يمكننا استخدام نموذج الانحدار الخطي للمربعات الصغرى Least Square Linear Regression. يظهر في الرسم البياني التالي البيانات وتوقعات النموذج:

```
df.plot.scatter(0, 1, s=50);
plot_curve(curves[0])
```

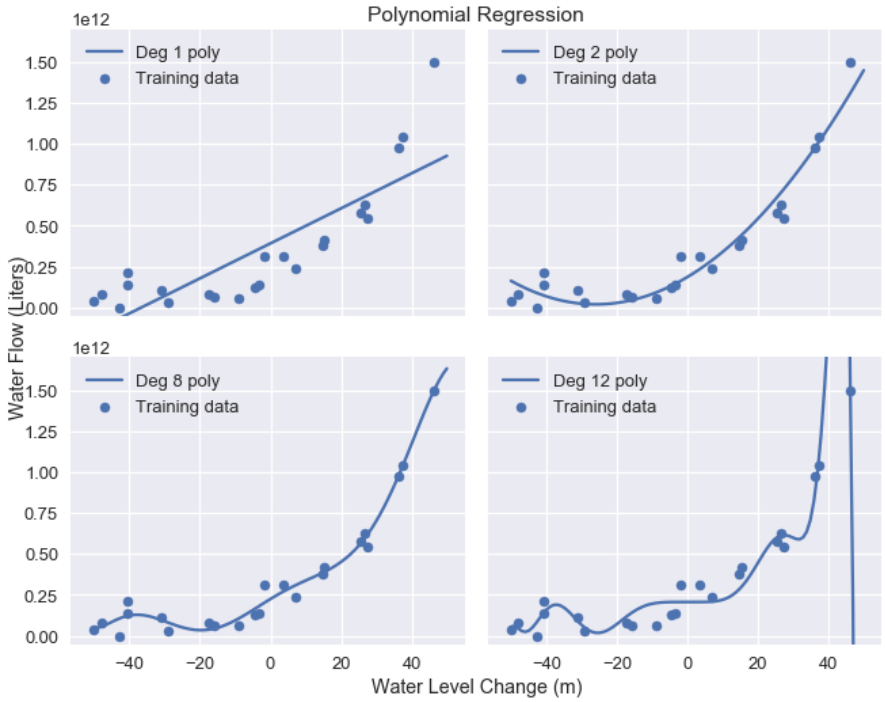


في الكود البرمجي السابق إستخدم الكاتب الدالة plot_curve والمصفوفة curves للمساعدة في الرسم، وعرفها كالتالي مع دالة مساعدة أخرى make_curve:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from collections import namedtuple

Curve = namedtuple('Curve', ['xs', 'ys'])

def make_curve(clf, x_start=-50, x_end=50):
    xs = np.linspace(x_start, x_end, num=100)
    ys = clf.predict(xs.reshape(-1, 1))
    return Curve(xs, ys)

def plot_curve(curve, ax=plt, **kwargs):
    ax.plot(curve.xs, curve.ys, **kwargs)

X = df.iloc[:, [0]].values
y = df.iloc[:, 1].values

degrees = [1, 2, 8, 12]
clfs = [Pipeline([('poly', PolynomialFeatures(degree=deg, include_bias=False)),
                  ('reg', LinearRegression())])
        .fit(X, y)
        for deg in degrees]

curves = [make_curve(clf) for clf in clfs]
```

يظهر في الرسم البياني السابق أن هذا النموذج لا يغطي كامل النقاط في نمط هذه البيانات؛ والنموذج لدية انحياز عالي. كما فعلنا في السابق، يمكننا حل هذه المشكلة بإضافة خصائص متعددة الحدود للبيانات. نقوم بإضافة خصائص من الدرجة 2، 8 و 12؛ الرسوم البيانية التالية تظهر نتائج تدريب البيانات لكل نموذج:

```
plot_curves(curves)
```



إستخدم الكاتب دالة `plot_curves` لنفس المصفوفة المستخدمة في المثال السابق `curves` للمساعدة في رسم النماذج، وعرفها كالتالي مع مزيد من الدوال المساعدة:

```
def flatten(seq): return [item for subseq in seq for item in subseq]

def plot_data(df=df, ax=plt, **kwargs):
    ax.scatter(df.iloc[:, 0], df.iloc[:, 1], s=50, **kwargs)

def plot_curves(curves, cols=2):
    rows = int(np.ceil(len(curves) / cols))
    fig, axes = plt.subplots(rows, cols, figsize=(10, 8),
                             sharex=True, sharey=True)
    for ax, curve, deg in zip(flatten(axes), curves, degrees):
        plot_data(ax=ax, label='Training data')
        plot_curve(curve, ax=ax, label=f'Deg {deg} poly')
        ax.set_ylim(-5e10, 170e10)
        ax.legend()

    # إضافة الرسوم الكبيرة وإخفاء الإطارات
    fig.add_subplot(111, frameon=False)
    # إخفاء الحدود والعلامات والعناوين في الرسم
    plt.tick_params(labelcolor='none', top='off', bottom='off',
                    left='off', right='off')
    plt.grid(False)
    plt.title('Polynomial Regression')
    plt.xlabel('Water Level Change (m)')
    plt.ylabel('Water Flow (Liters)')
    plt.tight_layout()
```

كما هو متوقع، متعددة الحدود من الدرجة الـ 12 طابقت بيانات التدريب بشكل مناسب ولكن يبدو أن النموذج مضبوط بنمط زائف على البيانات بسبب التشويش. يظهر ذلك نموذج آخر للمقايضة بين الانحياز والتباين؛ النموذج الخطي لدية انحياز عالي وتباين قليل بينما متعددة الحدود من الدرجة 12 لديها انحياز قليل وتبيان عالي.

التحقق من المعاملات

عند التحقق من معاملات نموذج متعددة الحدود من الدرجة الـ 12 يظهر لنا أن النموذج يقوم بالتوقع بناءًا على المعادلة التالية:

$$207097470825 + 1.8x + 482.6x^2 + 601.5x^3 + 872.8x^4 + 150486.6x^5$$
$$+2156.7x^6 - 307.2x^7 - 4.6x^8 + 0.2x^9 + 0.003x^{10} - 0.00005x^{11} + 0x^{12}$$

فيها x هي قيمة تغير مستوى المياه في ذلك اليوم.

معاملات النموذج كبيرة جداً، خاصة لمتعددات الحدود ذات الدرجات العالية والتي تساهم بشكل عالي في تباين النموذج (مثلاً x^5 و x^6).

مغلّمات المعاقبة

لنتذكر أن النموذج الخطي يقوم بالتوقعات بناءاً على التالي، θ هي وزن النموذج و x هي متّجهة الخصائص:

$$f_{\hat{\theta}}(x) = \hat{\theta} \cdot x$$

لضبط النموذج، نقوم بتقليل دالة التكلفة لمتوسط الخطأ التربيعي، فيها X تمثل مصفوفة البيانات و y هي النتائج التي اطلعت عليها:

$$L(\hat{\theta}, X, y) = \frac{1}{n}\sum_i (y_i - f_{\hat{\theta}}(X_i))^2$$

لتقليل التكلفة في المعادلة السابقة، نقوم بضبط $\hat{\theta}$ حتى نصل لأفضل قيم للأوزان Weights أياً كان حجمها. لكن، وجدنا أن كلما كان الوزن عالي وبخصائص أكثر تعقيداً ينتج لنا تباين عالي للنموذج. على العكس، إذا كان بإمكاننا التعديل في دالة التكلفة لمعاقبة الأوزان العالية، سيكون النموذج الناتج ذو تباين قليل. نستخدم الضبط للقيام بذلك.

الضبط L2: انحدار Ridge

في هذا الجزء سنتعرف على طريقة الضبط L_2، طريقة لمعاقبة الأوزان الكبيرة في دوال التكلفة للتقليل من تباين النموذج. راجعنا بشكل مبسط الانحدار الخطي، ثم تعرفنا على الضبط كأداة للتعديل في دالة التكلفة.

لتطبيق الانحدار الخطي للمربعات الصغرى، نستخدم النموذج:

$$f_{\hat{\theta}}(x) = \hat{\theta} \cdot x$$

نقوم بضبط النموذج عن طريق التقليل دالة تكلفة متوسط الخطأ التربيعي:

$$L(\hat{\theta}, X, y) = \frac{1}{n}\sum_i^n (y_i - f_{\hat{\theta}}(X_i))^2$$

في المعادلة السابقة، X تمثل مصفوفة البيانات $n \times p$، و x تمثل سطر من أسطر X، و y تمثل النتائج التي أطلع عليها، و $\hat{\theta}$ هي وزن النموذج.

التعريف

لإضافة الضبط L_2 إلى النموذج، نقوم بتعديل دالة التكلفة السابقة:

$$L(\hat{\theta}, X, y) = \frac{1}{n}\sum_i (y_i - f_{\hat{\theta}}(X_i))^2 + \lambda \sum_{j=1}^{p} {\hat{\theta}_j}^2$$

نلاحظ أن دالة التكلفة مشابهة للسابقة مع إضافة الضبط L_2 في المعادلة $\lambda \sum_{j=1}^{p} {\hat{\theta}_j}^2$. المجموع ($\sum$) في هذه المعادلة يقوم بجمع تربيع كل وزن في النموذج $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_p$. يظهر لنا في المعادلة أيضاً رمز جديد لمتغير رقمي في النموذج λ والذي يتم تعديله لتحديد قيمة **عقوبة الضبط Regularization Penalty**.

معادلة الضبط الذي أضفناها تسبب زيادة في التكلفة إذا كانت القيم في $\hat{\theta}$ بعيدة عن صفر. مع إضافة الضبط، الوزن للنموذج المثالي يقلل من الخسارة ومعاقبة الضبط بدلاً من الخسارة فقط. بما أن نتائج أوزان النموذج عادة ما تكون قليلة، فسيكون النموذج ذو تبيان قليل وانحياز عالي.

استخدام الضبط L_2 مع نموذج خطي ودالة التكلفة لمتوسط الخطأ التربيعي يطلق عليه **انحدار Ridge**.

متغيرات الضبط

يتحكم متغير الضبط λ بعقوبة الضبط. عندما تكون λ قليلة ينتج قيمة عقاب قليل، إذا كانت $\lambda = 0$ تكون قيمة معاقبة الضبط أيضاً 0 وبالتالي لم يتم ضبط دالة التكلفة.

عندما تكون λ عالية ينتج لنا قيمة عقاب عاليه وبالتالي يكون النموذج بسيط. الزيادة في λ تقلل من التباين وتزيد من الانحياز في النموذج. نستخدم التحقق المتقاطع لاختيار قيمة λ التي تقلل من خطأ التحقق.

ملاحظة عن الضبط في مكتبة `scikit-learn`:

توفر مكتبة `scikit-learn` نموذج الانحدار ومضاف إليها بشكل تلقائي الضبط. مثلاً، لتطبيق انحدار Ridge يمكنك استخدام النموذج الخطي `sklearn.linear_model.Ridge`. لاحظ أن جميع النماذج في مكتبة `scikit-learn` تطلق على متغير الضبط `alpha` بدلاً من λ.

توفر مكتبة `scikit-learn` بشكل بسيط ومناسب نماذج مضبوطة تقوم بتطبيق التحقق المتقاطع لاختيار قيمة مناسبة لـ λ. مثلاً، `sklearn.linear_model.RidgeCV` يسمح للمستخدم إدخال قيم متغيرات الضبط وبشكل أوتوماتيكي تقوم بتطبيق التحقق المتقاطع لاختيار أفضل قيمة للمتغير والذي يحصل على أقل خطأ في بيانات التحقق.

استبعاد مصطلح الانحياز

لاحظ أن رمز الانحياز θ_0 لم يتم ضمة لعملية الجمع في معادلة الضبط. لا نقوم بمعاقبة الانحياز لأن الزيادة في انحياز النموذج لا تؤدي للزيادة في التباين؛ ببساطة يقوم الانحياز بتغير جميع التوقعات إلى قيم ثابتة.

تسوية البيانات

لاحظ أن معادلة الضبط $\lambda \sum_{j=1}^{p} \hat{\theta}_j^{\,2}$ تعاقب كل $\hat{\theta}_j$ بشكل متكافئ. ولكن، تأثير كل قيمة من $\hat{\theta}_j$ يختلف بناءاً على البيانات نفسها. لنأخذ مثلاً هذه الجزء من بيانات تدفق المياه بعد إضافة متغيرات متعددة الحدود من الدرجة الثامنة:

```
pd.DataFrame(clfs[2].named_steps['poly'].transform(X[:5]),
             columns=[f'deg_{n}_feat' for n in range(8)])
```

	deg_0_feat	deg_1_feat	...	deg_6_feat	deg_7_feat
0	15.94-	253.98	...	261095791.08-	4161020472.12
1	29.15-	849.9	...	17897014961.65-	521751305227.70
2	36.19	1309.68	...	81298431147.09	2942153527269.12
3	37.49	1405.66	...	104132296999.30	3904147586408.71
4	48.06-	2309.65	...	592123531634.12-	28456763821657.70

```
5 rows × 8 columns
```

نلاحظ أن متعددة الحدود من الدرجة السابعة حصلت على قيم أعلى بكثير عن متعددة الحدود من الدرجة الأولى. يعني ذلك أن الوزن العالي في نموذج متعددة الحدود من الدرجة السابعة أثر على التوقعات أكثر من الوزن عالي لنموذج متعددة الحدود من الدرجة الأولى. إذا قمنا بتطبيق الضبط على هذه البيانات مباشرة، ستقوم معاقبة الضبط بتقليل الوزن للنموذج بشكل غير مناسب للخصائص في الدرجات الصغرى. بشكل عملي، ينتج عن ذلك تباين عالي للنموذج حتى بعد تطبيق الضبط لأن الخصائص ذات التأثير الأكبر على التوقعات لن تتأثر.

لحل ذلك، نقوم بـ **التسوية Normalization** جميع الأعمدة عن طريق طرح المتوسط منها وتغير القيم فيها إلى شكل مدرج Scale بين -1 و 1. في مكتبة `scikit-learn` أغلب نماذج الانحدار تسمح بتطبيق التسوية باستخدام المتغير `normalize=True` للقيام بتسوية البيانات قبل الضبط.

طريقة أخرى مشابهة للتسوية هي **التوحيد Standardization** لأعمدة البيانات عن طريق طرح المتوسط وتقسيمه على الانحراف المعياري لكل عمود.

استخدام انحدار Ridge

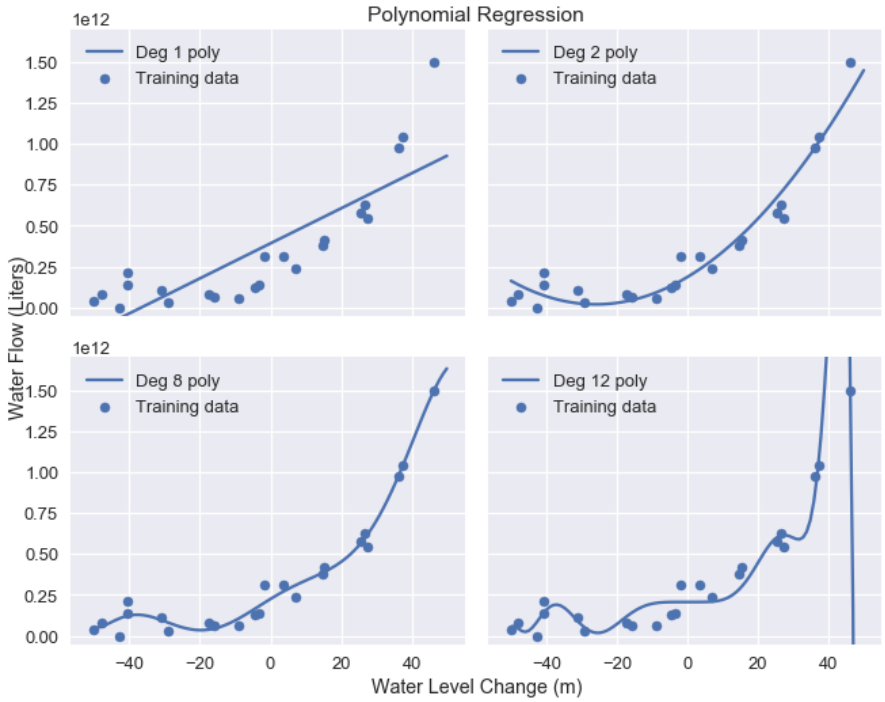
استخدمنا سابقاً خصائص متعددة الحدود لضبط متعددات الحدود من الدرجة 2، 8 و 12 لبيانات تدفق المياه. البيانات الأساسية ونتائج توقعات النموذج كالتالي:

```
df
```

	water_level_change	water_flow
0	15.94-	60422330445.52
1	29.15-	33214896575.60
2	36.19	972706380901.06
...	...	...
20	7.09	236352046523.78
21	46.28	1494256381086.73
22	14.61	378146284247.97

```
23 rows × 2 columns
```

```
plot_curves(curves)
```



لتطبيق انحدار Ridge، نقوم أولاً باستخراج مصفوفة البيانات و متّجهة النتائج من البيانات:

```
X = df.iloc[:, [0]].values
y = df.iloc[:, 1].values
print('X: ')
print(X)
print()
print('y: ')
print(y)
```

```
X:
[[-15.94]
 [-29.15]
 [ 36.19]
 ...
 [  7.09]
 [ 46.28]
 [ 14.61]]

y:
[6.04e+10 3.32e+10 9.73e+11 ... 2.36e+11 1.49e+12 3.78e+11]
```

ثم نقوم بتحويل القيم في X إلى متعددة الحدود من الدرجة 12:

```
from sklearn.preprocessing import PolynomialFeatures

# نريد تحديد include_bias=False
# لأن مصنفات مكتبة sklearns
# تقوم بشكل تلقائي بإضافة قيم الانحياز

X_poly_8 = PolynomialFeatures(degree=8, include_bias=False).fit_transform(X)
print('First two rows of transformed X:')
print(X_poly_8[0:2])
```

```
First two rows of transformed X:
[[-1.59e+01  2.54e+02 -4.05e+03  6.45e+04 -1.03e+06  1.64e+07 -2.61e+08
   4.16e+09]
 [-2.92e+01  8.50e+02 -2.48e+04  7.22e+05 -2.11e+07  6.14e+08 -1.79e+10
   5.22e+11]]
```

نقوم بتحديد قيم alpha التي ستختارها مكتبة scikit-learn باستخدام التحقق المتقاطع، ثم نستخدم نموذج RidgeCV لضبط البيانات بعد التحويل:

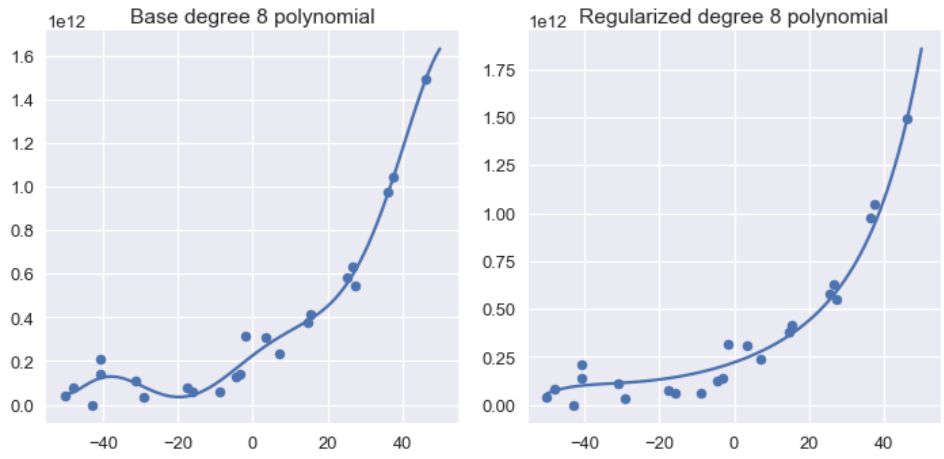
```
from sklearn.linear_model import RidgeCV
```

```
from sklearn.linear_model import RidgeCV

alphas = [0.01, 0.1, 1.0, 10.0]

# تذكر ان تحدد normalize=True

# ليتم تسوية البيانات
clf = RidgeCV(alphas=alphas, normalize=True).fit(X_poly_8, y)

# إظهار قيمة Alpha
clf.alpha_
```

```
0.1
```

أخيراً، نقوم برسم نتائج التوقعات لنموذج متعددة الحدود من الدرجة الثامنة بجانب النموذج من الدرجة الثامنة الذي تم ضبطه:

```
fig = plt.figure(figsize=(10, 5))

plt.subplot(121)
plot_data()
plot_curve(curves[2])
plt.title('Base degree 8 polynomial')

plt.subplot(122)
plot_data()
plot_curve(ridge_curves[2])
plt.title('Regularized degree 8 polynomial')
plt.tight_layout()
```



إستخدم الكاتب نفس الطريقة السابقة لرسم انحدار Ridge، عن طريق إنشاء مصفوفة أسماها ridge_curves وعرفها كالتالي:

```
from sklearn.linear_model import Ridge

ridge_clfs = [Pipeline([('poly', PolynomialFeatures(degree=deg, include_bias=False)),
                        ('reg', Ridge(alpha=0.1, normalize=True))])
        .fit(X, y)
        for deg in degrees]

ridge_curves = [make_curve(clf) for clf in ridge_clfs]
```

نلاحظ أن متعددة الحدود التي تم ضبطها منحناها أكثر سلاسة وتمر على أغلب نقاط البيانات.

بمقارنة المعامِلات للنموذج الذي تم ضبطه والذي لم يتم ضبطه نلاحظ أن انحدار Ridge يفضل وضح وزن للنموذج على متعددات الحدود ذو الدرجات الدنيا:

```
base = coef_table(clfs[2]).rename(columns={'Coefficient Value': 'Base'})
ridge = coef_table(ridge_clfs[2]).rename(columns={'Coefficient Value': 'Regularized'})

pd.options.display.max_rows = 20
display(base.join(ridge))
pd.options.display.max_rows = 7
```

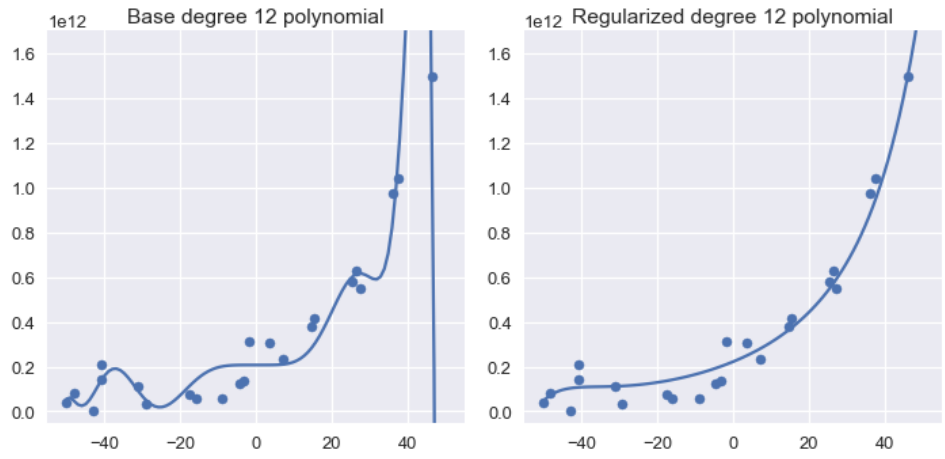
	Base	Regularized
degree		
0	225782472111.94	221063525725.23
1	13115217770.78	6846139065.96
2	144725749.98-	146158037.96
3	10355082.91-	1930090.04
4	567935.23	38240.62
5	9805.14	564.21
6	249.64-	7.25
7	2.09-	0.18
8	0.03	0.00

استخدم الكاتب دالتين `coef_table` و `coefs` تقومان بإنشاء المُعامِلات على شكل جدول في DataFrame، وتعريفها كالتالي:

```
def coefs(clf):
    reg = clf.named_steps['reg']
    return np.append(reg.intercept_, reg.coef_)

def coef_table(clf):
    vals = coefs(clf)
    return (pd.DataFrame({'Coefficient Value': vals})
            .rename_axis('degree'))
```

إعادة تطبيق نفس العملية على متعددة الحدود من الدرجة 12 يظهر نتائج مماثلة:

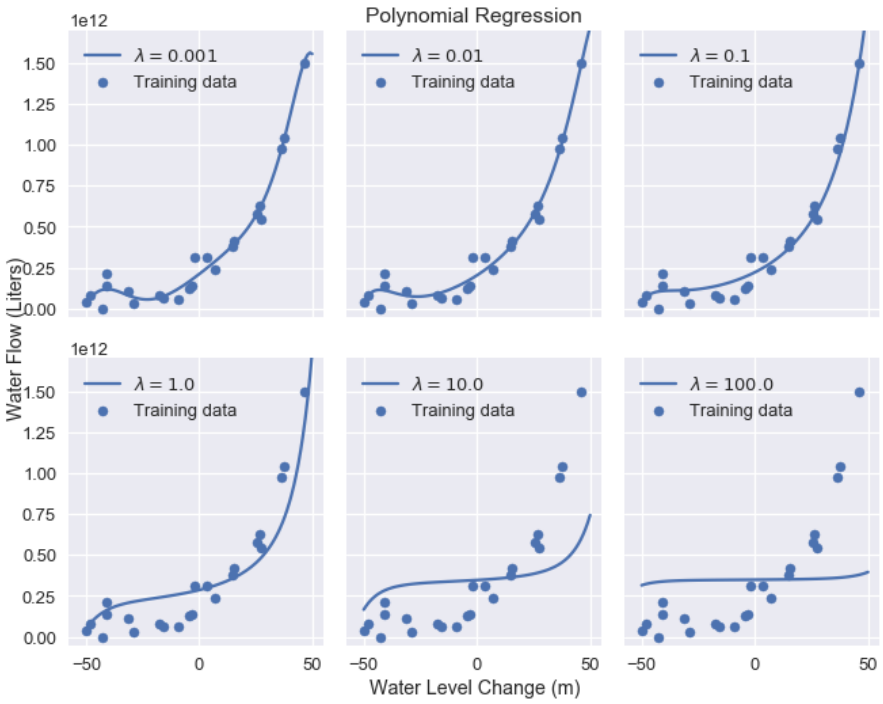
```
fig = plt.figure(figsize=(10, 5))

plt.subplot(121)
plot_data()
plot_curve(curves[3])
plt.title('Base degree 12 polynomial')
plt.ylim(-5e10, 170e10)

plt.subplot(122)
plot_data()
plot_curve(ridge_curves[3])
plt.title('Regularized degree 12 polynomial')
plt.ylim(-5e10, 170e10)
plt.tight_layout()
```



نحصل على نموذج أكثر بساطة كلما قمنا في الزيادة في متغير الضبط. الرسم البياني التالي يوضح تأثير الزيادة في قيمة الضبط من 0.001 حتى 100.0:

```
alphas = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]

alpha_clfs = [Pipeline([
    ('poly', PolynomialFeatures(degree=12, include_bias=False)),
    ('reg', Ridge(alpha=alpha, normalize=True))]
).fit(X, y) for alpha in alphas]

alpha_curves = [make_curve(clf) for clf in alpha_clfs]
```

```
labels = [f'$\\lambda = {alpha}$' for alpha in alphas]

plot_curves(alpha_curves, cols=3, labels=labels)
```



إستخدمت الدالة `plot_curves` للمساعدة على رسم المنحنيات، وتعريفها كالتالي:

```
def plot_curves(curves, cols=2, labels=None):
    if labels is None:
        labels = [f'Deg {deg} poly' for deg in degrees]
    rows = int(np.ceil(len(curves) / cols))
    fig, axes = plt.subplots(rows, cols, figsize=(10, 8),
                             sharex=True, sharey=True)
    for ax, curve, label in zip(flatten(axes), curves, labels):
        plot_data(ax=ax, label='Training data')
        plot_curve(curve, ax=ax, label=label)
        ax.set_ylim(-5e10, 170e10)
        ax.legend()

    # إضافة الرسوم الكبيرة وإخفاء الإطارات
    fig.add_subplot(111, frameon=False)
    # إخفاء الحدود والعلامات والعناوين في الرسم
    plt.tick_params(labelcolor='none', top='off', bottom='off',
                    left='off', right='off')
    plt.grid(False)
    plt.title('Polynomial Regression')
    plt.xlabel('Water Level Change (m)')
    plt.ylabel('Water Flow (Liters)')
    plt.tight_layout()
```

كما نرى، الزيادة في متغير الضبط تزيد من انحياز النموذج. إذا كان متغير الضبط عالي جداً، يصبح النموذج نموذجاً ثابتاً لأن أي قيمة لوزن النموذج غير صفر ينتج عنها عقاب عالي.

ملخص انحدار Ridge

استخدام ضبط L_2 يسمح لنا بضبط انحياز وتباين النموذج عن طريق عقاب أوزان النموذج العالية. ضبط L_2 للانحدار الخطي للمربعات الصغرى يعرف أيضاً باسم انحدار Ridge. استخدام الضبط يضيف متغير إضافي للنموذج يرمز له λ والذي نقوم بضبطه باستخدام التحقق المتقاطع.

الضبط L1: انحدار Lasso

في هذا الجزء سنتعرف على طريقة الضبط L_1، طريقة أخرى من طرق الضبط تفيدنا في اختيار الخصائص.

سنبدأ بمراجعة بسيطة للضبط L_2 للانحدار الخطي. استخدمنا النموذج:

$$f_\theta(x) = \hat{\theta} \cdot x$$

قمنا بضبط النموذج عن طريق تقليل دالة التكلفة لمتوسط الخطأ التربيعي وأضفنا إليها الضبط:

$$L(\hat{\theta}, X, y) = \frac{1}{n}\sum_i (y_i - f_{\hat{\theta}}(X_i))^2 + \lambda \sum_{j=1}^{p} \hat{\theta}_j^{\,2}$$

في المعادلة السابقة، X تمثل مصفوفة البيانات $n \times p$، و x تمثل سطر من أسطر X، و y تمثل النتائج التي أطلع عليها، و $\hat{\theta}$ هي وزن النموذج، و λ تمثل قيمة متغير الضبط.

تعريف الضبط

لإضافة الضبط L_1 إلى النموذج، نقوم بتعديل دالة التكلفة السابقة إلى:

$$L(\hat{\theta}, X, y) = \frac{1}{n}\sum_i (y_i - f_{\hat{\theta}}(X_i))^2 + \lambda \sum_{j=1}^{p} |\hat{\theta}_j|$$

لاحظ أن الفرق بين المعادلتين فقط في جزئية الضبط. الضبط L_1 يعاقب مجموع الأوزان المطلقة بدلاً من مجموع التربيع.

استخدام الضبط L_2 مع نموذج خطي ودالة التكلفة لمتوسط الخطأ التربيعي يطلق عليه **انحدار** Lasso) .**Lasso** هي اختصار ل Least Absolute Shrinkage and Selection Operator معامل الانكماش والاختيار المطلق الأقل.) 📝

المقارنة بين انحدار Ridge و Lasso

لتطبيق انحدار Lasso، سنستخدم النموذج LassoCV من مكتبة `scikit-learn`، وهي نسخة من نموذج Lasso التي تطبق التحقق المتقاطع لاختيار متغيرات الضبط. في الأسفل، نستعرض بيانات تدفق المياه من السد:

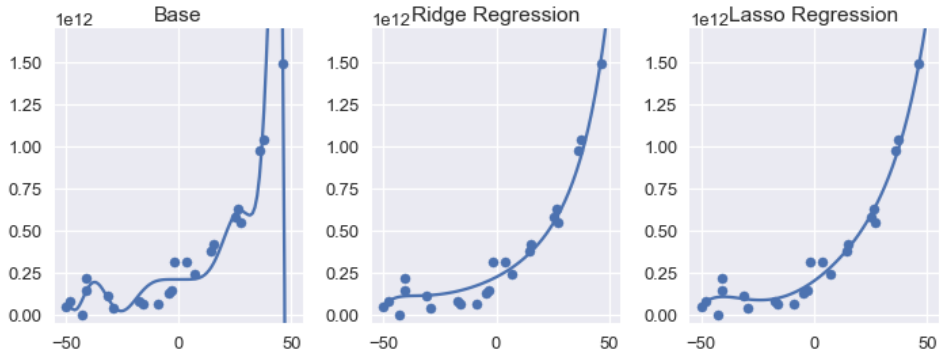
```
df
```

	water_level_change	water_flow
0	15.94-	60422330445.52
1	29.15-	33214896575.60
2	36.19	972706380901.06
...	...	...
20	7.09	236352046523.78
21	46.28	1494256381086.73
22	14.61	378146284247.97

```
23 rows × 2 columns
```

بما أن الخطوات في كلا الطريقتين متطابقة، سنستعرض النتائج لجميع النماذج معاً مع متعددة الحدود من الدرجة 12.:

```
fig = plt.figure(figsize=(10, 4))

plt.subplot(131)
plot_data()
plot_curve(curves[3])
plt.title('Base')
plt.ylim(-5e10, 170e10)

plt.subplot(132)
plot_data()
plot_curve(ridge_curves[3])
plt.title('Ridge Regression')
plt.ylim(-5e10, 170e10)

plt.subplot(133)
plot_data()
plot_curve(lasso_curves[3])
plt.title('Lasso Regression')
plt.ylim(-5e10, 170e10)
plt.tight_layout()
```



إستخدم الكاتب نفس الطريقة السابقة لرسم انحدار Lasso، عن طريق إنشاء مصفوفة أسماها `lasso_curves` وعرفها كالتالي:

```
from sklearn.linear_model import LassoCV

lasso_clfs = [Pipeline([('poly', PolynomialFeatures(degree=deg, include_bias=False)),
                        ('reg', LassoCV(normalize=True, precompute=True, tol=0.001))])
       .fit(X, y)
       for deg in degrees]
lasso_curves = [make_curve(clf) for clf in lasso_clfs]
```

نلاحظ ان كلا النموذجين المضبوطة لديها تباين قليل مقارنة مع متعددة الحدود من الدرجة 12. نلاحظ أن استخدام الضبط L_2 أو L_1 ينتج لنا نتائج قريبة جداً أن تكون متطابقة. لكن، عند مقارنة المُعاملات لنموذجي انحدار Ridge و Lasso، يظهر لنا أهم الفروقات بين النوعين من الضبط: نموذج انحدار Lasso حدد بعض من أوزان النموذج إلى القيمة صفر:

```
ridge = coef_table(ridge_clfs[3]).rename(columns={'Coefficient Value': 'Ridge'})
lasso = coef_table(lasso_clfs[3]).rename(columns={'Coefficient Value': 'Lasso'})

pd.options.display.max_rows = 20
pd.set_option('display.float_format', '{:.10f}'.format)
display(ridge.join(lasso))
pd.options.display.max_rows = 7
pd.set_option('display.float_format', '{:.2f}'.format)
```

	Ridge	Lasso
degree		
0	221303288116	198212062407
1	6953405308	9655088668
2	142621063.9	198852674.2
3	1893283.057	0
4	38202.15203	34434.34589
5	484.4262914	975.6965959
6	8.152512652	0
7	0.1197232472	0.0887942172
8	0.0012506185	0
9	0.0000289599	0
10	0.0000000004-	0
11	0.0000000069	0
12	0.0000000001-	0

عند العودة للنتائج في الجدول، نلاحظ أن انحدار Ridge كانت نتائجه غير صفريه للأوزان في جميع خصائص متعددة الحدود. بينما انحدار Lasso، كانت لدية نتائج تساوي صفر في سبعة من الخصائص.

بمعنى آخر، أن نموذج انحدار Lasso تجاهل بشكل كامل الكثير من الخصائص عند القيام بالتوقعات. ولكن، الرسم البياني السابق أظهر أن نموذج انحدار Lasso قام بتوقعات مطابقة بشكل كبير لنموذج انحدار Ridge.

اختيار الخصائص مع انحدار Lasso

عندما يقوم انحدار Lasso بتطبيق **اختيار الخصائص Feature Selection**، يتجاهل أجزاء من الخصائص الأصلية عند ضبط الخصائص بالنموذج. هذه الطريقة مفيدة عندما نعمل على بيانات عالية الأبعاد بخصائص كثيره. نموذج يستخدم القليل من الخصائص للقيام بالتوقعات سيعمل بشكل

أسرع من نموذج آخر يحتاج الكثير من عمليات الحساب. بما أن الخصائص غير المرغوب فيها تزيد من تباين النموذج دون التقليل من الانحياز، يمكننا أحياناً زيادة دقة النماذج عن طريق استخدام انحدار Lasso ليقوم باختيار جزء من الخصائص ليستخدمها.

Ridge × Lasso عملياً

إذا كان هدفنا فقط الوصول إلى أعلى دقة في التوقع، يمكننا تجربة كلا النوعين من الضبط مع التحقق المتقاطع والاختيار بينهما.

أحياناً نفضل نوع من الضبط عن الآخر لأنها يطابق بشكل كثير المجال الذي نعمل عليه. مثلاً، إذا كنا نعرف أن ظاهره ما والتي نحاول أن نبني لها نموذج تنتج من العديد من العوامل الصغيرة، فربما نفضل استخدام انحدار Ridge لأنه لن يتجاهل أحد هذه العوامل الصغيرة. من ناحية أخرى، بعض النتائج تأتي جزء الخصائص شديدة التأثير. نفضل استخدام انحدار Lasso في تلك الحالات لأنه يتجاهل الخصائص غير المرغوب فيها.

ملخص انحدار Lasso

استخدام الضبط L_1، كما في الضبط L_2، يساعدنا على ضبط انحياز وتباين النموذج عن طريق معاقبة الأوزان العالية في النموذج. الضبط L_1 لانحدار خطي ودالة التكلفة لمتوسط الخطأ التربيعي يطلق عليه بالمصطلح الشائع انحدار Lasso. يمكن أن ستخدم انحدار Lasso في اختيار الخصائص بما أنه يتجاهل الخصائص غير المهمة.

التصنيف

مقدمة

حتى الآن ألقينا نظرة على نماذج للانحدار، طريقة للقيام بتوقعات مستمرة وبالأرقام بناءًا على البيانات. الآن ننتقل إلى **التصنيف Classification**، وهي طريقة للقيام بتوقعات تصنيفية بناءًا على البيانات. مثلاً، قنوات توقعات الطقس مهتمة بتوقع ما إذا سيكون غداً يوم ماطر أو غير ذلك بناءًا على حال الطقس اليوم.

معاً، التصنيف والانحدار تعتبر من النماذج الأولية والتي يَتَّجِه لها في *التعليم الموجَّه Supervised Learning*، والذي فيه يتعلم النموذج عن طريق تمرير البيانات ونتائجها عليه.

يمكننا إعادة تشكيل التصنيف على أنه نوع من أنواع الانحدار. بدلاً من أن نبني نموذج ليتوقع أرقام، ننشأ نموذج ليتوقع احتمالية أن قيمة من البيانات تنتمي لصنف ما. يسمح لنا ذلك بإعادة استخدام تقنيات الانحدار الخطي في الانحدار على الاحتمالات أو ما يسمى **الانحدار اللوجستي Logistic Regression**. 📝

الانحدار في الاحتمالات

في كرة السلة، تحسب نقاط المباريات عندما يقوم اللاعب بإدخال الكره في السلة. أحد اللاعبين، ليبرون جيمس، يعرف بأنه أحد أفضل من لعب كرة السلة بسبب قدراته في تسجيل النقاط.

يلعب ليبرون في الدوري الأمريكي لكرة السلة للمحترفين (NBA). قمنا بجمع جميع محاولات التسجيل لليبرون في المباريات الإقصائية لعام 2017 باستخدام موقع stats.nba.com.

لتحميل البيانات lebron.csv اضغط هنا.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
sns.set()
sns.set_context('talk')
np.set_printoptions(threshold=20, precision=2, suppress=True)
pd.options.display.max_rows = 7
pd.options.display.max_columns = 8
pd.set_option('precision', 2)

lebron = pd.read_csv('lebron.csv')
lebron
```

	game_date	minute	opponent	action_type	shot_type	shot_distance	shot_made
0	20170415	10	IND	Driving Layup Shot	2PT Field Goal	0	0
1	20170415	11	IND	Driving Layup Shot	2PT Field Goal	0	1
2	20170415	14	IND	Layup Shot	2PT Field Goal	0	1
...	...	...	...	...	...	...	...
381	20170612	46	GSW	Driving Layup Shot	2PT Field Goal	1	1
382	20170612	47	GSW	Turnaround Fadeaway shot	2PT Field Goal	14	0
383	20170612	48	GSW	Driving Layup Shot	2PT Field Goal	2	1

384 rows × 7 columns

كل سطر في هذه البيانات يحتوي على المعلومات التالية عن محاولات التسجيل لليرون جيمس:

- `game_date`: تاريخ المباراة.
- `minute`: الدقيقة التي حاول فيها التسجيل (مدة كل مباراة في كرة السلة الأمريكية 48 دقيقة).
- `opponent`: اختصار اسم الفريق المنافس.
- `action_type`: الطريقة التي تمت فيها محاولة التسجيل.
- `shot_type`: نوع الرمية (إما رمية نقطتين أو ثلاث نقاط).
- `shot_distance`: مسافة ليبرون عن السلة عندما قام بمحاولة التسجيل.
- `shot_made`: تكون 0 عندما تكون محاولة التسجيل فاشلة و 1 عندما تكون محاولة التسجيل ناجحة.

نريد أن نستخدم هذه البيانات لإجراء التوقعات عن احتمالية تسجيل ليبرون للمزيد من النقاط. تعتبر هذه مشكلة *تصنيف Classification*؛ نتوقع صِنف، وليس رقم كما نفعل في الانحدار>

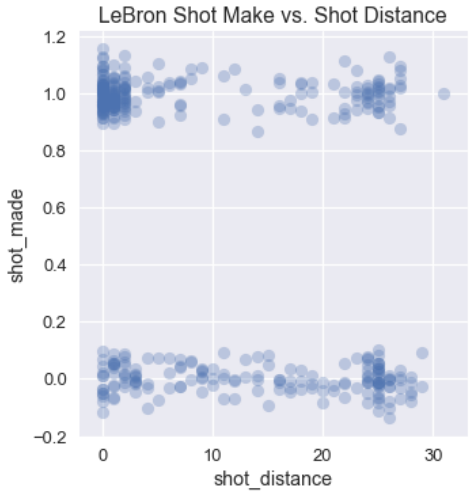
يمكننا إعادة صياغة مشكلة التصنيف هذه إلى مشكلة انحدار نتوقع فيها *الاحتمالية Probability* إذا كانت الكرة ستدخل في السلة أم لا. مثلاً، نتوقع أن محاولات ليبرون للتسجيل ستخطأ السلة عندما تكون المحاولة من مسافة بعيدة.

نقوم برسم محاولات التسجيل، نظهر فيها المسافة عن السلة في المحور x و ما إذا تم تسجيل تلك المحاولة أو لا في المحور y.

```
sns.lmplot(x='shot_distance', y='shot_made',
           data=jitter_df(lebron, 'shot_distance', 'shot_made'),
           fit_reg=False,
           scatter_kws={'alpha': 0.3})
plt.title('LeBron Shot Make vs. Shot Distance');
```

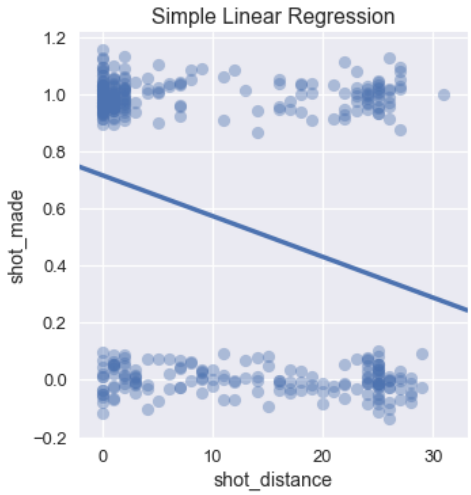
إستخدم الكاتب الدالة `jutter_df` لتحويل البيانات إلى قيم عشوائية عن طريق إضافة قيمة عشوائية لها، وعرفها كالتالي:

```
def jitter_df(df, x_col, y_col):
   x_jittered = df[x_col] + np.random.normal(scale=0, size=len(df))
   y_jittered = df[y_col] + np.random.normal(scale=0.05, size=len(df))
   return df.assign(**{x_col: x_jittered, y_col: y_jittered})
```



يمكن أن نرى أن ليبرون يسجل أكثر عندما يكون بحوالي 5 أقدام أو قل إلى السلة. عند ضبط نموذج بسيط للانحدار الخطي في المربعات الصغرى على هذه البيانات ينتج لنا التوقعات التالية:

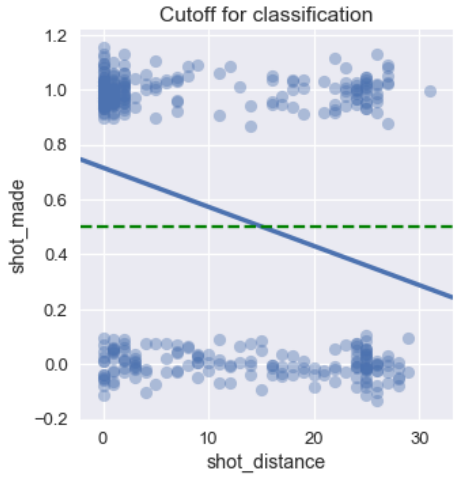
```
sns.lmplot(x='shot_distance', y='shot_made',
           data=jitter_df(lebron, 'shot_distance', 'shot_made'),
           ci=None,
           scatter_kws={'alpha': 0.4})
plt.title('Simple Linear Regression');
```



يستخدم الانحدار الخطي لتوقع قيم رقمية. ولكن عندما نريد تطبيقه على التصنيف، نريد تحويل هذه القيم الرقمية إلى تصنيف: تسجيل الكرة أو لا. يمكننا تطبيق ذلك عن طريق تحديد خط للقطع، أو **حد فصل التصنيف Classification Threshold**. إذا توقع الانحدار قيمة أكبر من 0.5، فيعني ذلك أن التوقع هو أن الكرة ستدخل السلة. على عكس ذلك، إذا توقع أقل من 0.5 فإن الكرة ستخطأ السلة.

رسمنا حد الفصل في الرسم البياني باللون الأخضر المتقطع. بناءًا على هذا الحد الفاصل، فإن نموذجنا يتوقع أن ليبرون سيجل الكرة إذا كان على بعد 15 قدم أو أقل عن السلة.

```
sns.lmplot(x='shot_distance', y='shot_made',
           data=jitter_df(lebron, 'shot_distance', 'shot_made'),
           ci=None,
           scatter_kws={'alpha': 0.4})
plt.axhline(y=0.5, linestyle='--', c='g')
plt.title('Cutoff for Classification');
```



في الخطوات السابقة، حاولنا تطبيق الانحدار لتوقع احتمالية أن كرة ستصيب الهدف. إذا كان انحدارنا ينتج احتمالية، فإن تحديد الحد الفاصل على 0.5 يعني أننا نوقع أن الكرة ستصيب الهدف عندما يتوقع النموذج قيمة أعلى من ذلك. سنعود لموضوع حد فصل التصنيف في جزء آخر في هذا الفصل.

مشاكل الانحدار الخطي في الاحتمالات

للأسف، لا يمكننا اعتماد نتائج نموذجنا الخطي على أنها احتمالات. الاحتمالات الصحيحة يجب أن تكون بين 0 و 1، ولكن نموذجنا الخطي يخالف هذا الشرط. مثلاً، احتمالية أن يسجل ليبرون كرة من على بعد 100 قدم عن السلة يجب أن تكون أقرب إلى الصفر. ولكن، في حالتنا هنا، النموذج سيتوقع قيمة سلبية.

إذا قمنا بالتعديل على نموذجنا الخطي لتكون توقعاته عبارة عن احتمالات، لن نواجه مشكلة في استخدام توقعاته في التصنيف. يمكننا القيام بذلك باستخدام دالة توقع ودالة خسارة جديدة. يطلق على هذا النموذج بـ **الانحدار اللوجستي Logistic Regression**.

النموذج اللوجستي

في هذا الجزء، سنتعرف على **الانحدار اللوجستي**، نموذج خطي نستخدمة لتوقع الاحتمالات.

لنتذكر أن لضبط نموذج نحتاج لثلاث عناصر: نموذج يقوم بالتوقع، دالة خسارة، وطريقة لتحسين نتائج النموذج. سنستخدم النموذج الذي نعرفة الآن، الانحدار الخطي للمربعات الصغرى:

$$f_{\hat{\theta}}(\mathbf{x}) = \hat{\boldsymbol{\theta}} \cdot \mathbf{x}$$

ودالة الخسارة:

$$L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \frac{1}{n} \sum_i (y_i - f_{\boldsymbol{\theta}}(\mathbf{X}_i))^2$$

سنستخدم النزول الاشتقاقي كأداة لتحسين النتائج. في التعريف السابق، X تمثل مصفوفة البيانات $n \times p$ (وفيها n هي عدد البيانات و p عدد العناصر / الأعمدة)، تمثل $\mathbf{x}$ سطر من X، و y متّجه للنتائج التي سبق أن أطلع عليها. المتّجه $\hat{\boldsymbol{\theta}}$ الوزن المثالي للنموذج و $\boldsymbol{\theta}$ تمثل الوزن المتوسط الذي أنشئ أثناء محاولة تحسين النموذج.

الأعداد الحقيقية إلى احتمالات

لنرى أن نموذجنا $f_{\hat{\theta}}(\mathbf{x}) = \hat{\boldsymbol{\theta}} \cdot \mathbf{x}$ يمكن أن يتوقع أي رقم حقيقي $\mathbb{R}$ بما أنه ينتج مجموعة من الارقام خطية في $\mathbf{x}$، والتي بنفسها يمكن أن تحتوي على أي رقم من $\mathbb{R}$.

يمكننا بسهله رسم ذلك بيانياً عندما تكون x رقم متدرج. إذا كانت $\hat{\theta} = 0.5$ فأن النموذج سيكون $f_{\hat{\theta}}(\mathbf{x}) = 0.5x$. يمكن لتوقعات هذا النموذج أن تكون أي رقم من سالب مالا نهاية حتى موجب مالا نهاية:

```
xs = np.linspace(-100, 100, 100)
ys = 0.5 * xs
plt.plot(xs, ys)
plt.xlabel('$x$')
plt.ylabel(r'$f_\hat{\theta}(x)$')
plt.title(r'Model Predictions for $ \hat{\theta} = 0.5 $');
```

لمهام التصنيف، نريد تقييد $f_{\hat{\theta}}(\mathbf{x})$ بحيث تكون نتائجها عبارة عن احتمالية. يعني ذلك أن نتائجها تكون في المدى $[0, 1]$ أيضاً، نريد أن تتطابق القيم الكبرى في $f_{\hat{\theta}}(\mathbf{x})$ مع قيم كبرى في الاحتمالية والعكس للقيم الصغرى مع قيمة احتمالية صغرى.

الدالة اللوجستية

لإنجاز ذلك، سنتعرف على **الدالة اللوجستية Logistic Function**، ويطلق عليها في بعض الأحيان **الدالة السينية Sigmoid Function**:

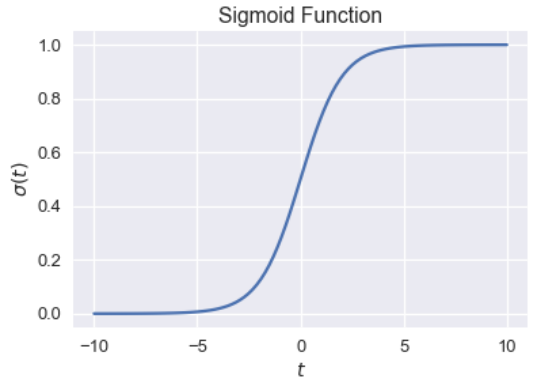
$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

لتسهيل القراءة، عادة ما نقوم بتغير e^x إلى $\exp(x)$:

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

قمنا برسم الدالة السينية للقيم التالية $t \in [-10, 10]$:

```
from scipy.special import expit
xs = np.linspace(-10, 10, 100)
ys = expit(xs)
plt.plot(xs, ys)
plt.title(r'Sigmoid Function')
plt.xlabel('$ t $')
plt.ylabel(r'$ \sigma(t) $');
```



نلاحظ أن الدالة السينية $\sigma(t)$ تأخذ رقم حقيقي $\mathbb{R}$ وتكون نتائجها فقط أرقام بين 0 و 1. تقوم الدالة تدريجياً بالصعود بناءًا على القيم المدخلة t؛ القيم الكبيرة في t هي القيم القريبة من 1، كما رغبناها أن تعمل. لم يتم ذلك بالصدفة؛ الدالة السينية يمكن اشتقاقها من نسبة لوغاريتميات الاحتمالات، ولكن قمنا بإخفاء الاشتقاق لتسهيل الشرح

تعريف النموذج اللوجستي

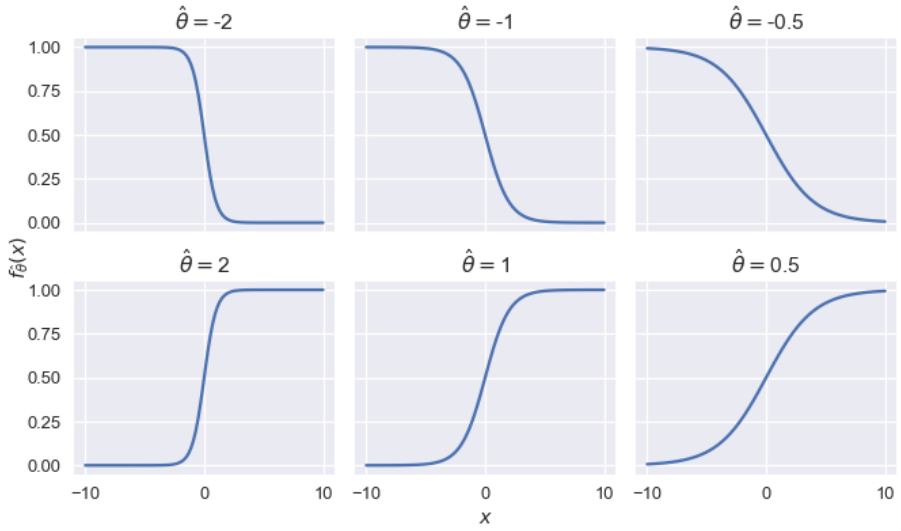
يمكننا الآن أخذ النموذج الخطي $\hat{\boldsymbol{\theta}} \cdot \mathbf{x}$ واستخدامه كمُدخل للدالة السينية لإنشاء **النموذج اللوجستي**:

$$f_{\hat{\theta}}(\mathbf{x}) = \sigma(\hat{\boldsymbol{\theta}} \cdot \mathbf{x})$$

بمعنى آخر، نأخذ نتيجة الانحدار الخطي، أي رقم حقيقي $\mathbb{R}$، ونستخدمه في الدالة السينية لتقييد نتائج النموذج النهائية لتكون نتيجة احتمالية صحيح كرقم بين صفر وواحد.

لمعرفة طريقة عمل النموذج اللوجستي بشكل بسيط، سنقوم بتقييد قيمة x لتكون رقم متدرج ورسم نتيجة النموذج اللوجستي لعدة قيم لـ $\hat{\theta}$:

```
# لا حاجة لفهم الكود البرمجي لأن الكاتب استخدمة لبناء الرسم البياني التوضيحي
def flatten(li): return [item for sub in li for item in sub]

thetas = [-2, -1, -0.5, 2, 1, 0.5]
xs = np.linspace(-10, 10, 100)

fig, axes = plt.subplots(2, 3, sharex=True, sharey=True, figsize=(10, 6))
for ax, theta in zip(flatten(axes), thetas):
    ys = expit(theta * xs)
    ax.plot(xs, ys)
    ax.set_title(r'$ \hat{\theta} = $' + str(theta))


fig.add_subplot(111, frameon=False)
plt.tick_params(labelcolor='none', top='off', bottom='off',
                left='off', right='off')
plt.grid(False)
plt.xlabel('$x$')
plt.ylabel(r'$ f_\hat{\theta}(x) $')
plt.tight_layout()
```



نلاحظ أن التغير في قيمة $\hat{\theta}$ يغير من حِدة المنحنى؛ كلما أبعدنا عن 0، كلما زادت حدة المنحنى. تغير الإشارة في $\hat{\theta}$ مع الإبقاء على نفس القيمة الرقمية ينتج لنا نفس النتائج لكن بشكل عكسي.

ملخص النموذج اللوجستي

تعرفنا على النموذج اللوجستي، طريقة جديدة للتوقع التي تُنتِج لنا توقعات كاحتمالات. لبناء النموذج، نستخدم مخرجات النموذج الخطي كمدخلات إلى الدالة اللوجستية غير الخطية.

دالة الخسارة للنموذج اللوجستي

قمنا بتعريف النموذج الخطي للاحتمالات، النموذج اللوجستي:

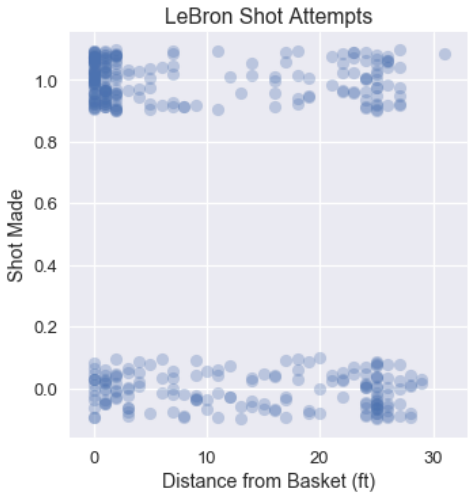
$$f_{\hat{\theta}}(\mathbf{x}) = \sigma(\hat{\boldsymbol{\theta}} \cdot \mathbf{x})$$

كما في النموذج الخطي، يحتوي النموذج على متغيرات $\hat{\boldsymbol{\theta}}$، مصفوفة تحتوي على متغير واحد لكل خاصية في $\mathbf{x}$. سنقوم الآن بحل مشكلة تعريف دالة الخسارة لهذا النموذج والتي ستسمح لنا بضبط متغيرات النموذج لهذه البيانات.

ما نريده هو أن يقوم النموذج بتوقعات مطابقة بشكل كبير للبيانات. في الأسفل قمنا برسم بياني لمحاولات التسجيل لليبرون في المباريات الإقصائية لعام 2017 باستخدام بعد كل تسديدة عن السلة:

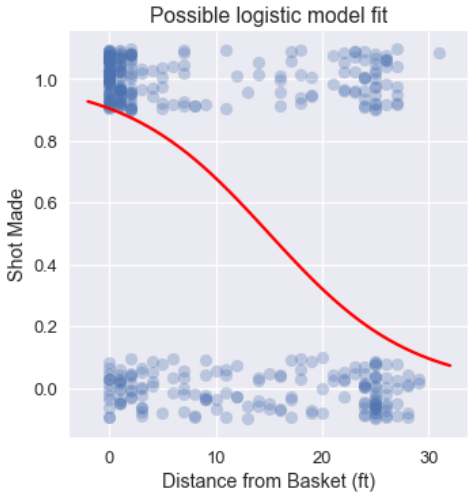
```
sns.lmplot(x='shot_distance', y='shot_made',
           data=lebron,
           fit_reg=False, ci=False,
           y_jitter=0.1,
           scatter_kws={'alpha': 0.3})
plt.title('LeBron Shot Attempts')
```

```
plt.xlabel('Distance from Basket (ft)')
plt.ylabel('Shot Made');
```



نلاحظ التجمع لكثير من المحاولات التي تم تسجيلها والتي كانت قريبة من السلة وتجمع قليل من البيانات لمحاولات تم تسجيلها من مسافات بعيدة عن السلة، نتوقع عندما نضبط النموذج اللوجستي على هذه البيانات أن النتيجة سيكون كالتالي:

```
from scipy.special import expit

sns.lmplot(x='shot_distance', y='shot_made',
           data=lebron,
           fit_reg=False, ci=False,
           y_jitter=0.1,
           scatter_kws={'alpha': 0.3})

xs = np.linspace(-2, 32, 100)
ys = expit(-0.15 * (xs - 15))
plt.plot(xs, ys, c='r', label='Logistic model')

plt.title('Possible logistic model fit')
plt.xlabel('Distance from Basket (ft)')
plt.ylabel('Shot Made');
```



على الرغم أن بإمكاننا استخدام دالة خسارة الخطأ التربيعي المتوسط كما فعلنا في نموذج الانحدار الخطي، هي ليست ملائمة للنموذج اللوجستي ويصعب تحسين نتائجها.

خسارة الانتروبيا التقاطعية

بدلاً من الخطأ التربيعي المتوسط، سنستخدم **خسارة الانتروبيا التقاطعية Cross-Entropy Loss**. لتمثل $\mathbf{X}$ مصفوفة البيانات $n \times p$، و $\mathbf{y}$ متّجه للنتائج التي سبق أن أطلع عليها، و $f_{\boldsymbol{\theta}}(\mathbf{x})$ تمثل النموذج اللوجستي. تحتوي $\boldsymbol{\theta}$ على قيم المتغيرات الحالية. باستخدام هذا التعريف، يمكننا تعريف معادلة متوسط خسارة الانتروبيا التقاطعية كالتالي: 📝

$$L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \frac{1}{n} \sum_i \left(-y_i \ln(f_{\boldsymbol{\theta}}(\mathbf{X}_i)) - (1 - y_i) \ln(1 - f_{\boldsymbol{\theta}}(\mathbf{X}_i))\right)$$

يمكن أن تلاحظ، كما فعلنا مسبقاً، أننا أخذنا متوسط الخسارة لكل نقطة معينة في البيانات. ما في داخل التعريف السابق يمثل الخسارة لنقطة معينة $(\mathbf{X}_i, y_i)$:

$$\ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i) = -y_i \ln(f_{\boldsymbol{\theta}}(\mathbf{X}_i)) - (1 - y_i) \ln(1 - f_{\boldsymbol{\theta}}(\mathbf{X}_i))$$

لنتذكر أن كل قيمة y_i ستكون إما 0 أو 1. إذا كانت $y_i = 0$، فأول مصطلح في الخسارة يساوي صفر. إذا كانت $y_i = 1$، فأن ثاني مصطلح في التعريف السابق يساوي صفر. لذا، لكل نقطة في بياناتنا، فإن فقط جزء واحد من تعريف دالة خسارة الانتروبيا التقاطعية يساهم في حساب قيمة الخسارة النهاية.

لنفترض أن $y_i = 0$ والاحتمالية المتوقعة كانت $f_{\boldsymbol{\theta}}(\mathbf{X}_i) = 0$، فإن نموذجنا كان توقعه صحيح. ستكون الخسارة لهذه النقطة كالتالي:

$$\begin{aligned}
\ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i) &= -y_i \ln(f_{\boldsymbol{\theta}}(\mathbf{X}_i)) - (1 - y_i) \ln(1 - f_{\boldsymbol{\theta}}(\mathbf{X}_i)) \\
&= -0 - (1 - 0) \ln(1 - 0) \\
&= -\ln(1) \\
&= 0
\end{aligned}$$

كما هو متوقع، الخسارة للتوقع الصحيح هي 0. يمكنك التأكد من ذلك عندما تكون نتائج توقع الاحتماليه أبعد من القيمة الحقيقة، فإن الخسارة تكون عالية.

التقليل من خسارة الانتروبيا التقاطعية يحتاج أن يقوم النموذج $f_{\boldsymbol{\theta}}(\mathbf{x})$ بأدق التوقعات الصحيحة الممكنة. لذلك، فإن دالة الخاسرة هذه مُحدبة، مما يجعل النزول الاشتقاقي مناسب لتحسين النتائج.

مشتقة خسارة الانتروبيا التقاطعية

لتطبيق النزول الاشتقاقي على خسارة الانتروبيا التقاطعية للنموذج يجب علينا حساب مشتقة الخسارة. أولاً، نقوم بحساب مشتقة الدالة السينية بما أننا سنستخدمها في عملية حساب الاشتقاق:

$$\begin{aligned}
\sigma(t) &= \frac{1}{1 + e^{-t}} \\
\sigma'(t) &= \frac{e^{-t}}{(1 + e^{-t})^2} \\
\sigma'(t) &= \frac{1}{1 + e^{-t}} \cdot \left(1 - \frac{1}{1 + e^{-t}}\right) \\
\sigma'(t) &= \sigma(t)(1 - \sigma(t))
\end{aligned}$$

يمكننا وصف مشتقة الدالة السينية من الدالة السينية نفسها.

للاختصار، نقوم بتعريف $\sigma_i = f_{\boldsymbol{\theta}}(\mathbf{X}_i) = \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})$. سنحتاج قريباً إلى مشتقة σ_i للمُتجهة $\boldsymbol{\theta}$ لذا سنقوم بحسابة الآن باستخدام قاعدة السلسلة: 📝

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \sigma_i &= \nabla_{\boldsymbol{\theta}} \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta}) \\
&= \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})(1 - \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})) \nabla_{\boldsymbol{\theta}} (\mathbf{X}_i \cdot \boldsymbol{\theta}) \\
&= \sigma_i (1 - \sigma_i) \mathbf{X}_i
\end{aligned}$$

والآن، سنوجد مشتقة خسارة الانتروبيا التقاطعية لمتغيرات النموذج $\boldsymbol{\theta}$. في تفصيل حل المشتقة في الأسفل، جعلنا $\sigma_i = f_{\boldsymbol{\theta}}(\mathbf{X}_i) = \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})$:

$$\begin{aligned}
L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) &= \frac{1}{n} \sum_i \left(-y_i \ln(f_{\boldsymbol{\theta}}(\mathbf{X}_i)) - (1 - y_i) \ln(1 - f_{\boldsymbol{\theta}}(\mathbf{X}_i))\right) \\
&= \frac{1}{n} \sum_i \left(-y_i \ln \sigma_i - (1 - y_i) \ln(1 - \sigma_i)\right) \\
\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) &= \frac{1}{n} \sum_i \left(-\frac{y_i}{\sigma_i} \nabla_{\boldsymbol{\theta}} \sigma_i + \frac{1 - y_i}{1 - \sigma_i} \nabla_{\boldsymbol{\theta}} \sigma_i\right) \\
&= -\frac{1}{n} \sum_i \left(\frac{y_i}{\sigma_i} - \frac{1 - y_i}{1 - \sigma_i}\right) \nabla_{\boldsymbol{\theta}} \sigma_i \\
&= -\frac{1}{n} \sum_i \left(\frac{y_i}{\sigma_i} - \frac{1 - y_i}{1 - \sigma_i}\right) \sigma_i (1 - \sigma_i) \mathbf{X}_i \\
&= -\frac{1}{n} \sum_i \left(y_i - \sigma_i\right) \mathbf{X}_i
\end{aligned}$$

التعريف البسيط للمشتقة يسمح لنا بضبط النموذج اللوجستي لخسارة الانتروبيا التقاطعية باستخدام النزول الاشتقاقي:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$$

لاحقاً سنتعمق ونحدث في المعادلة لأنواع مختلفة من النزول الاشتقاقي الدُفعات، العشوائي والدفعات الصغيرة.

ملخص خسارة الانتروبيا التقاطعية

بما أن دالة خسارة الانتروبيا التقاطعية مُحدبة، نقوم بتقليلها باستخدام النزول الاشتقاقي لضبط النموذج اللوجستي على البيانات. لدينا الآن العناصر المهمة للانحدار اللوجستي: النموذج، دالة الخسارة وطريقة لتحسين النتائج. في جزء لاحق سنتعمق أكثر لمعرفة سبب استخدامنا لمتوسط خسارة الانتروبيا التقاطعية في الانحدار اللوجستي

استخدام الانحدار اللوجستي

تعرفنا على جميع العناصر المطلوبة في الانحدار اللوجستي، أولاً، النموذج اللوجستي المستخدم لإجراء توقعات الاحتماليات:

$$f_{\hat{\theta}}(\mathbf{x}) = \sigma(\hat{\boldsymbol{\theta}} \cdot \mathbf{x})$$

ثم دالة خسارة الانتروبيا التقاطعية:

$$L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = = \frac{1}{n} \sum_i \left(-y_i \ln \sigma_i - (1 - y_i) \ln(1 - \sigma_i) \right)$$

وأخيراً، مشتقة خسارة الانتروبيا التقاطعية للنزول الاشتقاقي:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = -\frac{1}{n} \sum_i \left(y_i - \sigma_i \right) \mathbf{X}_i$$

في المعادلة السالفة، جعلنا $\mathbf{X}$ تمثل مصفوفة البيانات $n \times p$ ، تمثل $\mathbf{x}$ سطر من $\mathbf{X}$، و y متّجه للنتائج التي سبق أن أطلع عليها. و $f_{\hat{\theta}}(\mathbf{x})$ هي النموذج اللوجستي المثالي وذو متغيرات مثالية $\hat{\boldsymbol{\theta}}$. للاختصار، نعرف $\sigma_i = f_{\hat{\theta}}(\mathbf{X}_i) = \sigma(\mathbf{X}_i \cdot \hat{\boldsymbol{\theta}})$.

الانحدار اللوجستي على محاولات التسجيل لليبرون

لنعود للمشكلة التي واجهناها في بداية هذا الفصل: لنتوقع أي محاولات التسجيل لليبرون جيمس ستصيب الهدف. أولاً نحمل بيانات محاولات التصويب لليبرون في المباريات الإقصائية لعام 2017:

```
lebron = pd.read_csv('lebron.csv')
lebron
```

	game_date	minute	opponent	action_type	shot_type	shot_distance	shot_made
0	20170415	10	IND	Driving Layup Shot	2PT Field Goal	0	0
1	20170415	11	IND	Driving Layup Shot	2PT Field Goal	0	1
2	20170415	14	IND	Layup Shot	2PT Field Goal	0	1
...	...	...	...	...	...	...	...
381	20170612	46	GSW	Driving Layup Shot	2PT Field Goal	1	1
382	20170612	47	GSW	Turnaround Fadeaway shot	2PT Field Goal	14	0
383	20170612	48	GSW	Driving Layup Shot	2PT Field Goal	2	1

384 rows × 7 columns

لنلقي نظرة بشكل تفاعلي على هذه البيانات:

```
df_interact(lebron)
```

```
(384 rows, 7 columns) total
```

شُرحت هذه الدالة في فصل سابق ووظيفتها هي تمكين المستخدم من تصفح البيانات بشكل تفاعلي وتم تعريفها كالتالي:

```
%matplotlib inline
import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual

def df_interact(df, nrows=7, ncols=7):
   def peek(row=0, col=0):
       return df.iloc[row:row + nrows, col:col + ncols]
   if len(df.columns) <= ncols:
       interact(peek, row=(0, len(df) - nrows, nrows), col=fixed(0))
   else:
       interact(peek,
                row=(0, len(df) - nrows, nrows),
                col=(0, len(df.columns) - ncols))
   print('({} rows, {} columns) total'.format(df.shape[0], df.shape[1]))
```

نبدأ أولاً باستخدام مسافة محاولة التسجيل لتوقع ما إذا تم تسجيل المحاولة أم لا. توفر لنا مكتبة `scikit-learn` النموذج اللوجستي بشكل سهل عبر الكلاس `sklearn.linear_model.LogisticRegression`. لاستخدامه، نبدأ أولاً بإنشاء مصفوفة البيانات `X`، ومُتجهة النتائج التي المُطلع عليها `y`.

```
X = lebron[['shot_distance']].values
y = lebron['shot_made'].values
print('X:')
print(X)
print()
print('y:')
print(y)
```

```
X:
[[ 0]
 [ 0]
 [ 0]
 ...
 [ 1]
 [14]
 [ 2]]

y:
[0 1 1 ... 1 0 1]
```

وكالمعتاد، سنقسم بياناتنا إلى بيانات تدريب و اختبار:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=40, random_state=42
)
print(f'Training set size: {len(y_train)}')
print(f'Test set size: {len(y_test)}')
```

```
Training set size: 344
Test set size: 40
```

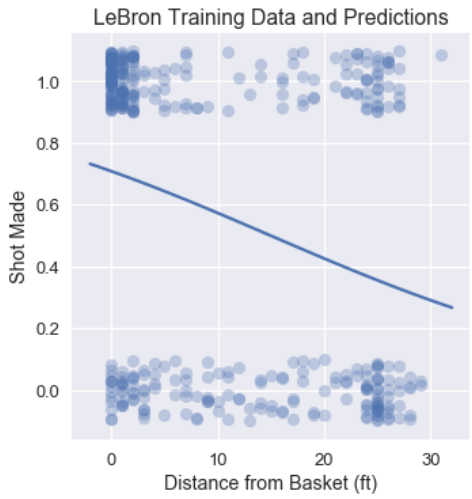
تجعل مكتبة scikit-learn إنشاء وضبط النموذج على X_train و y_train عملية سهلة جداً:

```
from sklearn.linear_model import LogisticRegression

simple_clf = LogisticRegression()
simple_clf.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

لعرض أداء النموذج في رسم بياني، سنتعرض النقاط الأصلية وتوقعات النموذج للاحتمالية:

```
sns.lmplot(x='shot_distance', y='shot_made',
           data=lebron,
           fit_reg=False, ci=False,
           y_jitter=0.1,
           scatter_kws={'alpha': 0.3})

xs = np.linspace(-2, 32, 100)
ys = simple_clf.predict_proba(xs.reshape(-1, 1))[:, 1]
plt.plot(xs, ys)

plt.title('LeBron Training Data and Predictions')
plt.xlabel('Distance from Basket (ft)')
plt.ylabel('Shot Made');
```



تقييم المصنف

إحدى الطرق لتقييم أداء النموذج هي بقياس دقة التوقعات: ما هي نسبة التوقعات الصحيحة؟

```
simple_clf.score(X_test, y_test)
```

```
0.6
```

يبدو أن أداء النموذج متدني قليلاً بدقة 0.6 على بيانات الاختبار. إذا قام نموذجنا بالتوقع على كل النقاط بشكل عشوائي، سنتوقع نتيجة الدقة ستكون 0.50. بالأصح، إذا قام نموذجنا ببساطة بتوقع كل محاولة للبيرون، سنحصل على نفس نسبة الدقة:

```
# حساب الدقة إذا كان التوقع 1
np.count_nonzero(y_test == 1) / len(y_test)
```

```
0.6
```

لهذا النموذج، استخدمنا متغير واحد من عدد مختلف من المتغيرات في هذه البيانات. في النموذج اللوجستي متعدد المتغيرات، سنحصل على نتائج أكثر دقة باستخدام أكثر من متغير.

نموذج لوجستي متعدد المتغيرات

الزيادة في عدد المتغيرات الرقمية في النموذج سهل جداً. على العكس، للمتغيرات من النوع النوعي/التصنيفية، نحتاج لتطبيق One-hot encoding. في الكود البرمجي التالي، أضفنا المزيد من المتغيرات للنموذج وهي `action_type`، `opponent`، `minute` و `shot_type` باستخدام كلاس `DictVectorizer` في مكتبة `scikit-learn` لتطبيق `One-hot encoding` على المتغيرات النوعية:

```
from sklearn.feature_extraction import DictVectorizer

columns = ['shot_distance', 'minute', 'action_type', 'shot_type', 'opponent']
rows = lebron[columns].to_dict(orient='row')

onehot = DictVectorizer(sparse=False).fit(rows)
X = onehot.transform(rows)
y = lebron['shot_made'].values

X.shape
```

```
(384, 42)
```

سنقوم مرة أخرى بتقسيم البيانات إلى بيانات تدريب و اختبار:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=40, random_state=42
)
print(f'Training set size: {len(y_train)}')
print(f'Test set size: {len(y_test)}')
```

```
Training set size: 344
Test set size: 40
```

أخيراً، نقوم بضبط النموذج مرة أخرى ونتحقق من الدقة:

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
print(f'Test set accuracy: {clf.score(X_test, y_test)}')
```

```
Test set accuracy: 0.725
```

دقة هذا النموذج أكثر بـ 12% من النموذج السابق الذي يستخدم متغير واحد. في جزء لاحق، سنستخدم مزيداً من الأدوات لتقييم أداء النموذج.

ملخص الانحدار اللوجستي

قمنا بتطوير عمليات حسابية رياضية وحاسوبية لاستخدام الانحدار اللوجستي في التصنيف. يستخدم الانحدار اللوجستي بشكل كبير لسهولته وفاعليته في التوقعات.

تقريب التوزيع الاحتمالي التجريبي

في هذا الجزء من الفصل، سنتعرف على **تباعد KL (KL divergence)** ونظهر كيف أن التقليل من متوسط التباعد KL في التصنيف ذو النتائج الثنائية (Binary 0/1) مماثل لاستخدام تقليل متوسط خسارة الانتروبيا التقاطعية. 📝

بما أن النتائج في الانحدار اللوجستي عبارة عن احتمالات، النموذج اللوجستي ينتج لنا نوعاً معين من توزيع الاحتمالات. بشكل خاص، بناءًا على المتغيرات المثالية $\hat{\theta}$، فأنه يقوم بتقييم احتمالية أن النتيجة y ستكون 1 لقيمة ادخلناها للنموذج $\mathbf{x}$.

مثلاً، لنفترض أن x قيمه رقمية تسجل احتمالية سقوط أمطار اليوم و $y = 1$ تعني أن السيد دو يحتاج لأخذ مظلته معه للعمل. النموذج اللوجستي ذو المتغيرات الرقمية $\hat{\theta}$ يتوقع احتمالية إحتياج السيد دو لأخذ مظلته معه إلى العمل بناءًا على توقع سقوط الأمطار هذا اليوم: $\hat{P}_\theta(y = 1|x)$.

جمع بيانات استخدام السيد دو لمظلته يوفر لنا طريقة لبناء توزيع إحتمالي تجريبي $P(y = 1|x)$. مثلاً، إذا كان هناك خمس أيام كانت فيها احتمالية أن حالة الطقس ستكون ماطر هي $x = 0.60$ وقام السيد دو بأخذ مظلته مره واحدة، فأن $P(y = 1|x = 0.60) = 0.20$. يمكننا حساب توزيع مماثل للاحتمالية لكل قيمة x تظهر في بياناتنا. بشكل طبيعي، بعد ضبط النموذج اللوجستي نريد ان تكون توقعات نموذجنا قريبة جداً إلى التوزيع الاحتمالي التجريبي في البيانات. يعني ذلك، لكل القيم x في البيانات، نريد:

$$\hat{P}_\theta(y = 1||x) \approx P(y = 1|x)$$

أحد طرق القياس الأكثر استخداماً لتحديد مدى قرب نتائج توزيعات الاحتمالان هو **تباعد كولباك - ليبلير Kullback–Leibler divergence** أو تباعد KL.

تعريف متوسط تباعد KL

يقوم تباعد KL بحساب الفرق بين توزيع احتمالية $\hat{P}_{\boldsymbol{\theta}}$ التي قام بحسابها النموذج اللوجستي بالمتغيرات $\boldsymbol{\theta}$ مع التوزيع الحقيقي للنقاط P في البيانات. يقوم بحساب مدى عدم دقة توقعات النموذج اللوجستي لتوزيعات البيانات.

تباعد KL لتصنيف ثنائي لتوزيعين P و $\hat{P}_{\boldsymbol{\theta}}$ لنقطة واحدة $(\mathbf{x}, y)$ هي:

$$D(P||\hat{P}_{\boldsymbol{\theta}}) = P(y = 0|\mathbf{x}) \ln\left(\frac{P(y = 0|\mathbf{x})}{\hat{P}_{\boldsymbol{\theta}}(y = 0|\mathbf{x})}\right) + P(y = 1|\mathbf{x}) \ln\left(\frac{P(y = 1|\mathbf{x})}{\hat{P}_{\boldsymbol{\theta}}(y = 1|\mathbf{x})}\right)$$

تباعد KL ليس متناسق، مثلاً تباعد $\hat{P}_{\boldsymbol{\theta}}$ عن P ليس نفس تباعد P عن $\hat{P}_{\boldsymbol{\theta}}$:

$$D(P||\hat{P}_{\boldsymbol{\theta}}) \neq D(\hat{P}_{\boldsymbol{\theta}}||P)$$

بما أن هدفنا هو استخدام $\hat{P}_{\boldsymbol{\theta}}$ للتوقع التقريبي لـ P، فنحن مهتمون بـ $D(P\|\hat{P}_{\boldsymbol{\theta}})$.

القيم المثالية لـ $\boldsymbol{\theta}$، والتي أشرنا لها بـ $\hat{\boldsymbol{\theta}}$، تقلل من متوسط تباعد KL لجميع نقاط البيانات n:

$$\text{Average KL Divergence} = \frac{1}{n}\sum_{i=1}^{n}\left(P(y_i = 0|\mathbf{X}_i) \ln\left(\frac{P(y_i = 0|\mathbf{X}_i)}{\hat{P}_{\boldsymbol{\theta}}(y_i = 0|\mathbf{X}_i)}\right) + P(y_i = 1|\mathbf{X}_i) \ln\left(\frac{P(y_i = 1|\mathbf{X}_i)}{\hat{P}_{\boldsymbol{\theta}}(y_i = 1|\mathbf{X}_i)}\right)\right)$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}}(\text{Average KL Divergence})$$

في المعادلة السابقة، النقطة i^{th} من نقاط البيانات أشير لها بـ $(\mathbf{X}_i, y_i)$ فيها $\mathbf{X}_i$ هي السطر i^{th} من البيانات $p \times n$ في مصفوفة البيانات $\mathbf{X}$ و y_i هي النتائج التي سبق أن أطلع عليها.

لا يعاقب تباعد KL القيم الشاذة والنادر حدوثها في P. إذا توقع النموذج احتمالية عالية لحدث نادر الحدوث، فإن كليهما $P(k)$ و $\ln\left(\frac{P(k)}{\hat{P}_{\boldsymbol{\theta}}(k)}\right)$ التباعد بينهما قليل. ولكن، إذا توقع النموذج احتمالية ضئيلة لحدث كثير الحدوث، فإن التباعد فيه عالي. يمكن أن نستنتج أن النموذج اللوجستي الذي يقوم بتوقعات صحيحة لأحداث كثيرة الحدوث لدية تباعد قليل من P على عكس نموذج يتوقع أحداث نادرة الحدود ولكن يختلف بشكل كبير في الأحداث كثيرة الحدوث.

استنتاج خسارة الانتروبيا التقاطعية من تباعد KL

تتشابه بشكل كبير المعادلة السابقة لتباعد KL مع خسارة الانتروبيا التقاطعية. سنعرض الآن في بعض العمليات الرياضية الجبرية أن تقليل متوسط تباعد KL يشابه تقليل متوسط خسارة الانتروبيا التقاطعية.

باستخدام خصائص اللوغاريتمات، يمكننا إعادة كتابة، يمكننا إعادة الكتابة كالتالي:

$$P(y_i = k|\mathbf{X}_i) \ln\left(\frac{P(y_i = k|\mathbf{X}_i)}{\hat{P}_{\boldsymbol{\theta}}(y_i = k|\mathbf{X}_i)}\right) = P(y_i = k|\mathbf{X}_i) \ln P(y_i = k|\mathbf{X}_i) - P(y_i = k|\mathbf{X}_i) \ln \hat{P}_{\boldsymbol{\theta}}(y_i = k|\mathbf{X}_i)$$

لاحظ بما أن المصطلح الأول لا يعتمد على $\boldsymbol{\theta}$، فأنه لا يؤثر على $\arg\min_{\boldsymbol{\theta}}$ ويمكن حذفه من المعادلة. المعادلة الناتجة بعد ذلك حي خسارة الانتروبيا التقاطعية للنموذج $\hat{P}_{\boldsymbol{\theta}}$:

$$\text{Average Cross-Entropy Loss} = \frac{1}{n}\sum_{i=1}^{n} -P(y_i = 0|\mathbf{X}_i) \ln \hat{P}_\theta(y_i = 0|\mathbf{X}_i) - P(y_i = 1|\mathbf{X}_i) \ln \hat{P}_\theta(y_i = 1|\mathbf{X}_i)$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta}(\text{Average Cross-Entropy Loss})$$

بما أن y_i هي قيمه معروفة، فإن احتمالية أن:

$$y_i = 1 \text{ و } P(y_i = 1|\mathbf{X}_i)$$

تساوي

$P(y_i = 0|\mathbf{X}_i)$ و y_i

هي $1 - y_i$.

توزيع احتماليات النموذج $\hat{P}_{\boldsymbol{\theta}}$ معطاة من نتائج الدالة السينية التي سبق أن تعرفنا عليها في جزئية سابقة. بعد القيام بتلك التعويضات في المعادلة، نصل إلى معادلة متوسط خسارة الانتروبيا التقاطعية:

$$\text{Average Cross-Entropy Loss} = \frac{1}{n}\sum_{i}\left(-y_i \ln(f_{\hat{\boldsymbol{\theta}}}(\mathbf{X}_i)) - (1-y_i)\ln(1-f_{\hat{\boldsymbol{\theta}}}(\mathbf{X}_i)\right)$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\theta}(\text{Average Cross-Entropy Loss})$$

التبرير الإحصائي لخسارة الانتروبيا التقاطعية

لدى خسارة الانتروبيا التقاطعية أيضاً أسس أساسية في علم الإحصاء. بما أن الانحدار اللوجستي يتوقع احتمالات، لو سُلم لنا نموذج لوجستي يمكننا أن نسأل، "ما هي احتمالية أن هذا النموذج يكون لنا تلك المجموعة من البيانات التي سبق أن اطلع عليها y؟" يمكننا بشكل طبيعي التعديل في المتغيرات للنموذج حتى تكون احتمالية حصولنا على بياناتنا كنتائج من النموذج عالية جداً. على الرغم من أننا لن نثبت ذلك في هذا الفصل، ولكن تطبيق ذلك مشابه للتقليل من خسارة الانتروبيا التقاطعية، هذا التفسير الإحصائي هو *أقصى احتمال maximum likelihood* لخسارة الانتروبيا التقاطعية.

ملخص تبرير خسارة الانتروبيا التقاطعية

يمكن القول ان متوسط تباعد KL هو متوسط الفرق اللوغارتمي بين توزيعان P و $\hat{P}_{\boldsymbol{\theta}}$ تم وزنها باستخدام P. التقليل من متوسط تباعد KL أيضاً يقلل من متوسط خسارة الانتروبيا التقاطعية. يمكننا التقليل من تباعد نموذج الانحدار اللوجستي عن طريق اختيار المتغيرات التي تصنف البيانات الشائعة بشكل صحيح.

ضبط النموذج اللوجستي

تحدثنا في جزء سابق عن **النزول الاشتقاقي بدفعات Batch Gradient Descent**، إحدى الخوارزميات التي تقوم بتحديث قيمة $\boldsymbol{\theta}$ بشكل متكرر حتى قيمة المتغير $\hat{\boldsymbol{\theta}}$ التي تقلل من الخسارة. و أيضاً تحدثنا عن **النزول الاشتقاقي العشوائي Stochastic Gradient Descent** و **النزول الاشتقاقي بدفعات صغيرة Mini-Batch Gradient Descent**، كُلها طُرق تستفيد من النظرية الإحصائية والحوسبة المُتوازية للتقليل من الوقت لتدريب خوارزمية النزول الاشتقاقي. في هذا الجزء، سنطبق هذه المفاهيم في الانحدار اللوجستي ونأخذ أمثلة على ذلك باستخدام دوال مكتبة `scikit-learn`.

النزول الاشتقاقي بدفعات

دالة التحديث العامة في النزول الاشتقاقي بدفعات معطاة كالتالي:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \cdot \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{(t)}, \mathbf{X}, \mathbf{y})$$

في الانحدار الخطي، نستخدم خسارة الانتروبيا التقاطعية كدالة خسارة:

$$L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}\left(-y_i \ln(f_{\boldsymbol{\theta}}(\mathbf{X}_i)) - (1-y_i)\ln(1-f_{\boldsymbol{\theta}}(\mathbf{X}_i))\right)$$

الاشتقاق لخسارة الانتروبيا التقاطعية هو $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma_i)\mathbf{X}_i$. تعويض ذلك في دالة التحديث يسمح لنا بإيجاد خوارزمية النزول الاشتقاقي الخاصة للانحدار اللوجستي. بجعل $\sigma_i = f_{\boldsymbol{\theta}}(\mathbf{X}_i) = \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})$:

$$\begin{aligned}\boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \alpha \cdot \left(-\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma_i\right)\mathbf{X}_i\right) \\ &= \boldsymbol{\theta}^{(t)} + \alpha \cdot \left(\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma_i\right)\mathbf{X}_i\right)\end{aligned}$$

- $\boldsymbol{\theta}^{(t)}$ هي القيمة الحالية المقدرة لـ $\boldsymbol{\theta}$ في التكرار t.
- α هي معدل التعلم.
- $-\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sigma_i\right)\mathbf{X}_i$ هي مشتقة خسارة الانتروبيا التقاطعية.
- $\boldsymbol{\theta}^{(t+1)}$ هي القيمة المقدرة التالية لـ $\boldsymbol{\theta}$ والتي تم حسابها بطرح نتيجة ضرب α و خسارة الانتروبيا التقاطعية التي تم حسابها في $\boldsymbol{\theta}^{(t)}$.

النزول الاشتقاقي العشوائي

يقرب النزول الاشتقاقي العشوائي من مشتقة دالة الخسارة في جميع النقاط باستخدام مستشفى الخسارة لنقطة واحدة. دالة التحديث العامة كالتالي، وفيها $\ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i)$ هي دالة الخسارة لنقطة واحدة: 📝

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i)$$

بالعودة للمثال في الانحدار اللوجستي، نقوم بتقريب مشتقة خسارة الانتروبيا التقاطعية لجميع النقاط باستخدام خسارة الانتروبيا التقاطعية لنقطة واحدة. يظهر ذلك في المعادلة التالية، والتي فيها $\sigma_i = f_{\boldsymbol{\theta}}(\mathbf{X}_i) = \sigma(\mathbf{X}_i \cdot \boldsymbol{\theta})$:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) &\approx \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i) \\ &= -(y_i - \sigma_i)\mathbf{X}_i \end{aligned}$$

عندما نقوم بتعويض التقريب إلى المعادلة العامة للنزول الاشتقاقي العشوائي، تتكون لنا دالة التحديث للنزول الاشتقاقي العشوائي للانحدار اللوجستي:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i) \\ &= \boldsymbol{\theta}^{(t)} + \alpha \cdot (y_i - \sigma_i)\mathbf{X}_i \end{aligned}$$

النزول الاشتقاقي بدفعات صغيرة

بشكل مماثل، يمكننا تقريب مشتقة خسارة الانتروبيا التقاطعية لجميع النقاط باستخدام عينة عشوائية من البيانات، تعرف أيضاً بالدفعات الصغيرة.

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) \approx \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \mathbf{X}_i, y_i)$$

نقوم بتعويض هذا التقريب لمشتقة خسارة الانتروبيا التقاطعية، ينتج عن ذلك دالة التحديث للنزول الاشتقاقي بدفعات صغيرة للانحدار اللوجستي:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \boldsymbol{\theta}^{(t)} - \alpha \cdot -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (y_i - \sigma_i)\mathbf{X}_i \\ &= \boldsymbol{\theta}^{(t)} + \alpha \cdot \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (y_i - \sigma_i)\mathbf{X}_i \end{aligned}$$

التطبيق في مكتبة Scikit-learn

الكلاس `SGDClassifier` في مكتبة `scikit-learn` يساعد على تنفيذ النزول الاشتقاقي العشوائي، يمكننا استخدامه بتحديد `loss=log`. بما أن `scikit-learn` لا تحتوي على نموذج لتطبيق النزول الاشتقاقي بدفعات صغيرة، سنقوم بمقارنة أداء `SGDClassifier` مع `LogisticRegression` على بيانات البريد الإلكتروني `emails_sgd.csv`. سنختصر جزئية اختيار الخصائص ولن نشرحها:

لتحميل البيانات emails_sgd.csv اضغط هنا.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier


# تحميل البيانات
emails = pd.read_csv('emails_sgd.csv').sample(frac=0.5)

# إختيار الخصائص ثم تحويلها لقيم رقمية
X, y = emails['email'], emails['spam']
X_tr = CountVectorizer().fit_transform(X)

# تقسيم البيانات لتدريب وإختبار
X_train, X_test, y_train, y_test = train_test_split(X_tr, y, random_state=42)

y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)

# ضبط النموذجان
log_reg = LogisticRegression(tol=0.0001, random_state=42)
stochastic_gd = SGDClassifier(tol=0.0001, loss='log', random_state=42)
```

```
%%time

from sklearn.metrics import accuracy_score, precision_score, recall_score

log_reg.fit(X_train, y_train)
log_reg_pred = log_reg.predict(X_test)
print('Logistic Regression')
print('  Accuracy:  ', accuracy_score(y_test, log_reg_pred))
print('  Precision: ', precision_score(y_test, log_reg_pred))
print('  Recall:    ', recall_score(y_test, log_reg_pred))
print()
```

```
Logistic Regression
  Accuracy:   0.9913793103448276
  Precision:  0.974169741697417
  Recall:     0.9924812030075187

CPU times: user 3.2 s, sys: 0 ns, total: 3.2 s
Wall time: 3.26 s
```

```
%%time

stochastic_gd.fit(X_train, y_train)
stochastic_gd_pred = stochastic_gd.predict(X_test)
print('Stochastic GD')
print('  Accuracy:  ', accuracy_score(y_test, stochastic_gd_pred))
print('  Precision: ', precision_score(y_test, stochastic_gd_pred))
print('  Recall:    ', recall_score(y_test, stochastic_gd_pred))
print()
```

```
Stochastic GD
  Accuracy:   0.9808429118773946
  Precision:  0.9392857142857143
  Recall:     0.9887218045112782

CPU times: user 93.8 ms, sys: 31.2 ms, total: 125 ms
Wall time: 119 ms
```

النتائج السابقة تظهر أن النموذج `SGDClassifier` استطاع إيجاد نتيجة في وقت أقل من `LogisticRegression`. على الرغم من أن نتائج تقييم النموذج `SGDClassifier` أسوأ قليلاً، إلا أننا نستطيع تحسين أداة النموذج `SGDClassifier` عبر ضبط الخصائص. أيضاً، هذه من المقايضات التي عادة ما يواجهها عالم البيانات. بناءًا على الوضع، يقوم عالم البيانات بإعطاء قيمة أعلى لسرعة إيجاد النتائج عن أداء

ملخص ضبط النموذج اللوجستي

يستخدم علماء البيانات النزول الاشتقاقي العشوائي للتقليل من تكلفة التشغيل والوقت. يمكننا مشاهدة ذلك في الانحدار اللوجستي، بما أن علينا فقط حساب مشتقة خسارة الانتروبيا التقاطعية لقيمة واحدة في كل تكرار بدلاً من كل القيم في النزول الاشتقاقي بدفعات. في المثال السابق عند استخدام `scikit-learn` والنموذج `SGDClassifier`، رأينا أن النزول الاشتقاقي العشوائي كانت نتائج أداءه أسوأ بعض الشيء، ولكن كأن أفضل من ناحية الوقت والسرعة بشكل كبير. في بيانات بحجم كبير و نماذج أكثر تعقيداً، الفرق في الوقت وسرعة إيجاد النتائج قد يكون أكبر بكثير وبذلك يكون أكثر أهمية.

تقييم النموذج اللوجستي

على الرغم من استخدامنا للدقة Accuracy لتقييم أداة النموذج اللوجستي في الجزء السابق، استخدام الدقة فقط يسبب بعض المشاكل التي سنتطرق لها في هذا الجزء من الفصل. للبدء في التحدث عن هذه المشاكل، سنتعرف على أداة أخرى لقياس أداء النموذج: **المساحة أسفل المنحنى Area Under Curve (AUC)**.

لنفترض أن لدينا بيانات 1000 بريد إلكتروني وتم تحديدها إذا كانت رسائل بريد مزعجة أم لا وهدفنا هو بناء نموذج ليفرق بين الرسائل الجديدة التي تصل لنا ما إذا كانت مزعجة أو لا. لنستعرض البيانات:

لتحميل البيانات selected_emails.csv اضغط هنا.

```
from sklearn.feature_extraction import DictVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import pandas as pd

emails=pd.read_csv('selected_emails.csv', index_col=0)

emails
```

	body	spam
0	…n Hi Folks,\n \n I’ve been trying to set a bu\	0
1	…Hah. I guess she doesn’t want everyone to kno	0
2	…This article from NYTimes.com \n has been sent	0
…	…	…

	body	spam
997	…html>\n <head>\n <meta http-equiv=”Conten>	1
998	…html>\n <head>\n </head>\n <body>\n \n <cente>	1
999	…n <html>\n \n <head>\n <meta http-equiv=3D”Co\	1

```
1000 rows × 2 columns
```

كل سطر يحتوي على محتوى رسالة البريد الإلكتروني في العمود `body` وما إذا كانت رسالة البريد مزعجة أو لا في العمود `spam`، فيه `0` يعني أن الرسالة سليمة و `1` يعني أن الرسالة مُزعجة.

لنبدأ بتقييم أداء 3 نماذج مختلفة:

- `ham_only`: يقيم كل رسالة بريد بأنها غير مزعجة “ham”.
- `spam_only`: يقيم كل رسالة بريد بأنها مزعجة “spam”.
- `words_list_model`: يتوقع الرسالة مزعجة أو لا بناءًا على توفر كلمات معينة في محتواها.

لنفترض أن لدينا مصفوفة للكلمات `words_list` نتوقع أنها تستخدم كثيراً في الرسائل المزعجة: “رجاءًا please”، “أضغط click”، “مال money”، “تجارة business”، و “حذف remove”. نبني نموذج `words_list_model` باستخدام الخطوات التالية: تحويل كل رسالة بريد إلى مصفوفة من الخصائص، بعد فصل الكلمات نحدد القيمة إلى 1 في حال وُجدت الكلمة في قائمة الكلمات `words_list` و 0 إذا لم تكن موجودة. مثلاً، عندما نستخدم الخمس كلمات السابقة وإذا وصلت لنا رسالة بريد إلكتروني محتواها “please remove by tomorrow”، فإن مصفوفة الخصائص الناتجة ستكون $[1, 0, 0, 0, 1]$. تطبيق ذلك ينتج لنا مصفوفة خصائص $\mathbf{X}$ ذات حجم `X 5 1000`.

الكود البرمجي التالي يظهر دقة النماذج. تم تجاهل شرح جزئية إنشاء النماذج وتدريبها للاختصار:

```
# تحديد كلمات الرسائل المزعجة
words_list = ['please', 'click', 'money', 'business', 'remove']

X = pd.DataFrame(words_in_texts(words_list, emails['body'].str.lower())).values
y = emails['spam'].values

# فصل البيانات لتدريب وإختبار
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=41, test_size=0.2
)

# ضبط النموذج
words_list_model = LogisticRegression(fit_intercept=True)
words_list_model.fit(X_train, y_train)

y_prediction_words_list = words_list_model.predict(X_test)
y_prediction_ham_only = np.zeros(len(y_test))
y_prediction_spam_only = np.ones(len(y_test))


print(f'ham_only test set accuracy: {np.round(accuracy_score(y_prediction_ham_only, y_test), 3)}')
print(f'spam_only test set accuracy: {np.round(accuracy_score(y_prediction_spam_only, y_test), 3)}')
print(f'words_list_model test set accuracy: {np.round(accuracy_score(y_prediction_words_list, y_test), 3)}')
```

```
ham_only test set accuracy: 0.96
spam_only test set accuracy: 0.04
words_list_model test set accuracy: 0.96
```

في المثال السابق، عرف الكاتب الدالة `words_in_texts` والتي تستقبل الكلمات والنص كمتغيرات على شكل مصفوفات، ونتيجتها مصفوفة من 0 و 1، 0 في حال لم تكن الكلمة موجودة في النص، و 1 في حال كانت موجودة:

```
def words_in_texts(words, texts):
   indicator_array = np.array([texts.str.contains(word) * 1 for word in words]).T

   return indicator_array
```

النموذج `words_list_model` كان تصنيفه صحيح بنسبة 96% على بيانات الاختبار. على الرغم من أن الدقة لهذا النموذج تبدو عالية، إلا أن النموذج `ham_only` حصل على نفس النتيجة عن طريق تحديد كل الرسائل كرسائل غير مزعجة. يسبب ذلك بعض المشاكل لأن النتائج هذه توضح أن بإمكاننا الحصول على نفس النتائج دون استخدام نموذج وفلترة الرسائل.

كما تظهر نتائج الدقة السابقة، قد تضللنا بعض نتائج أداء النماذج. يمكننا التأكد من دقة توقعات النموذج بشكل أدق باستخدام **مصفوفة الدقة Confusion Matrix**. مصفوفة الدقة لنموذج ثنائي النتائج (0/1) هي خريطة حرارية heatmap ذات مقاسات 2×2 تحتوي على نتائج التوقعات في محور و النتائج الحقيقية في المحور الآخر.

كل قيمة في مصفوفة الدقة عبارة عن النتائج المحتملة للنموذج. إذا تم إدخال رسالة بريد إلكتروني مزعجة إلى النموذج، فإن لدينا احتمالان:

- **True Positive (TP)** (القيمة اليسار في الأعلى): توقع النموذج كان صحيح للرسالة بتحديدها كمزعجة.

- **False Negative (FN)** (القيمة اليمين في الأعلى): توقع النموذج كان خاطئ بتوقعها كرسالة غير مزعجة، كونها من المفترض أن يتم تصنيفها كرسالة مزعجة. في حالتنا، توقع خاطئ يعني أن رسالة مزعجة تم تحديدها كرسالة غير مزعجة ووصلت إلى صندوق بريدنا الرئيسي Inbox.

بنفس الطريقة، إذا تم إدخال رسالة بريد ألكتروني غير مزعجة إلى النموذج، سيكون لدينا احتمالان:

- **False Positive (FP)** (القيمة اليسار في الأسفل): توقع النموذج كان خاطئ بتحديد الرسالة كرسالة مزعجة وهي عكس ذلك كونها مصنفة كغير مزعجة. في حالتنا، هذا التصنيف يعني أن الرسالة ستصل إلى البريد المزعج في بريدنا الإلكتروني ولن تصل للبريد الرئيسي Inbox.
- **True Negative (TN)** (القيمة اليمين في الأسفل): توقع النموذج كان صحيح للرسالة بتحديدها كغير مزعجة.

سبق أن شرحت هذة المفاهيم في تدوينة على الرابط التالي: هنا، يمكن إختصار الشرح في الجدول التالي:

Example of a confusion matrix مثال لنتائج مصفوفة الدقة		Predicted التوقع	
		Yes	No
Actual النتيجة الحقيقية	Yes	True Positive Yes توقع Yes والنتيجة الحقيقة	False Negative No توقع Yes والنتيجة الحقيقة
	No	False Positive Yes توقع No والنتيجة الحقيقة	True Negative No توقع No والنتيجة الحقيقة

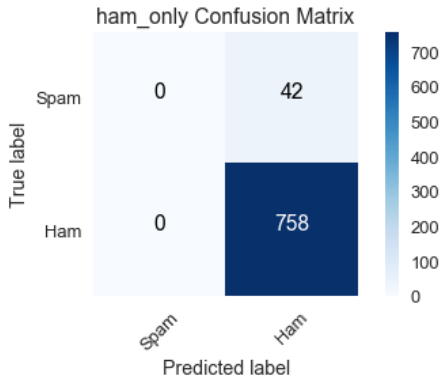
alioh.com

مع الاخذ بالإعتبار أن الشكل النهائي قد يختلف ويتم عكس الأعمدة Yes/No مثلاً.

تكلفة التوقع الخاطئ False Positive و False Negative تعتمد على ما نعمل عليه. في تصنيف رسائل البريد الإلكتروني، التصنيف الخاطئ False Positive يقوم بتحديد رسائل بريد إلكترونية مهمة كمزعجة وبالتالي يتم تجاهلها وعدم وضعها في البريد الرئيسي، لذا هي أسوأ من False Negative والذي فيه توضع الرسائل المزعجة في البريد الرئيسي. في المجال الطبي مثلاً، التوقع الخاطئ False Negative في تشخيص اختبار ما قد يكون أكثر أهمية من التوقع الخاطئ False Positive.

سنستخدم دالة مصفوفة الدقة Confusion Matrix في مكتبة `scikit-learn` لإنشاء مصفوفة الدقة للنماذج الثلاثة باستخدام بيانات التدريب. نتائج النموذج `ham_only` كالتالي:

```
from sklearn.metrics import confusion_matrix

class_names = ['Spam', 'Ham']

ham_only_cnf_matrix = confusion_matrix(y_train, ham_only_y_pred, labels=[1, 0])

plot_confusion_matrix(ham_only_cnf_matrix, classes=class_names,
                      title='ham_only Confusion Matrix')
```



استخدم الكاتب الدالة `plot_confusion_matrix` للمساعدة على الرسم والمتغير `ham_only_y_pred` لتحديد جميع القيم كغير مزعجة وعرفهما كالتالي:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    ترسم هذه الدالة مصفوفة الدقة
    يمكن تبسيط النتائج باستخدام المتغير `normalize=True`.
    """
    import itertools
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.grid(False)

ham_only_y_pred = np.zeros(len(y_train))
spam_only_y_pred = np.ones(len(y_train))
words_list_model_y_pred = words_list_model.predict(X_train)
```

لنجمع من الرسم البياني السابق مجموع النتائج في الصفوف لبيانات التدريب ونرى إذا تم تصنيفها في التصنيف الذي تنتمي إليه:

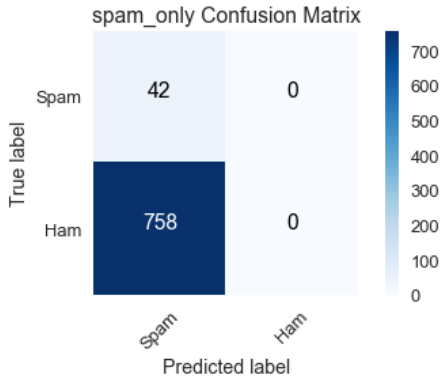
- True label = spam (السطر الأول): مجموع الرسائل التي تم توقعها كمزعجة وهي فعلاً مزعجة (True Positive 0)، والنتائج التي تم توقعها كغير مزعجة وهي في الحقيقة مزعجة (False Negative 42)، يظهر لذلك أن هناك 42 رسالة بريد إلكتروني مزعجة في بيانات التدريب.
- True label = ham (السطر الثاني): مجموع الرسائل التي توقعها كمزعجة وهي في الحقيقة غير مزعجة (False Positive 0)، ويليها توقعات الرسائل التي كانت غير مزعجة وهي فعلاً غير مزعجة (True Negative 758)، يعني ذلك أن هناك 758 رسالة بريد غير مزعجة في بيانات التدريب.

لنجمع من الرسم البياني السابق مجموع النتائج في الأعمدة لبيانات التدريب ونرى أداء النموذج في توقع تصنيفها الذي تنتمي إليه:

- Predicted label = spam (العمود الأول): مجموع توقعات الرسائل التي تم توقعها كمزعجة وهي فعلاً مزعجة (True Positive 0)، مجموع توقعات الرسائل التي توقعها كمزعجة وهي في الحقيقة غير مزعجة (False Positive 0) يظهر أن توقعات `ham_only` أنه لا يوجد أي رسالة بريد إلكتروني مزعجة في بيانات التدريب.
- Predicted label = ham (العمود الثاني): مجموع التوقعات التي تم توقعها كغير مزعجة وهي في الحقيقة مزعجة (False Negative 42)، مجموع توقعات الرسائل التي كانت غير مزعجة وهي فعلاً غير مزعجة (True Negative 758) يظهر أن توقعات `ham_only` أن هناك 800 رسالة بريد إلكتروني غير مزعجة في بيانات التدريب.

نرى أن النموذج `ham_only` لدية دقة عالية $\left(\frac{758}{800} \approx .95\right)$ لأن هناك 758 رسالة بريد إلكتروني غير مزعجة في بيانات التدريب من أصل 800.

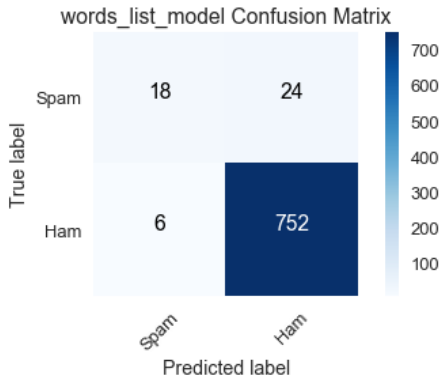
```
spam_only_cnf_matrix = confusion_matrix(y_train, spam_only_y_pred, labels=[1, 0])

plot_confusion_matrix(spam_only_cnf_matrix, classes=class_names,
                      title='spam_only Confusion Matrix')
```



في الجانب الآخر، النموذج `spam_only` كانت توقعاته أن بيانات التدريب لا تحتوي على أي رسالة بريد إلكتروني غير مزعجة `ham`، وكما هو واضح في مصفوفة الدقة النتيجة أبعد ما تكون عن الحقيقة والتي فيها 758 رسالة بريد إلكتروني غير مزعجة.

لنلقي نظرة على النموذج الذي يهمنا `words_list_model` ونتيجة مصفوفة الدقة:

```
words_list_model_cnf_matrix = confusion_matrix(y_train, words_list_model_y_pred, labels=[1, 0])

plot_confusion_matrix(words_list_model_cnf_matrix, classes=class_names,
                      title='words_list_model Confusion Matrix')
```



مجموع الصفوف مطابق لتلك في النتائج في مصفوفات الدقة للنماذج `ham_only` و `spam_only` كما هو متوقع لأن النتائج الحقيقية للمجموع في بيانات التدريب لن تتغير مهما تغير النموذج.

من بين ال 42 رسالة بريد مزعج، توقع النموذج `words_list_model` بشكل صحيح 18 رسالة، وهذا أداء سيئ. دقته العالية كانت بسبب العدد الكبير المتوقع للنتائج الصحيحة للرسائل الغير مزعجة True Negative، يعتبر ذلك غير كافٍ لأنه لا يفي بالغرض في فلترة رسائل البريد المزعجة.

بيانات رسائل البريد الإلكتروني هي مثال على **بيانات ذات تصنيف غير المتوازن Class-imbalanced dataset**، فيها الكثير من التصنيفات تنتمي لتصنيف أكثر من الآخر. في حالة بيانات البريد الإلكتروني، فإن الكثير من الرسائل تم تصنيفها كغير مزعجة. مثال آخر منتشر هو في بيانات اكتشاف الأمراض عندما يكون انتشار المرض في المجتمع الإحصائي قليل. نسبة الحصول على نتيجة أن المريض ليس لدية المرض عند إجراء الفحص الطبي أعلى سبب أن الكثير من المرضى ليس لديهم المرض، عدم قدرته على كشف المرضى بذلك المرض يظهر عدم فائدة النموذج. 📝

ننتقل الآن إلى الحساسية والنوعية، طرق قياس مناسبة للبيانات ذات التصنيف غير المتوازن.

الحساسية

تسمى **الحساسية Sensitivity** أيضاً بـ **معدل التوقع والنتيجة الإيجابيان True Positive Rate** يقيس نسبة البيانات التي تنتمي إلى التصنيف الإيجابي وتوقعها النموذج بشكل صحيح: 📝

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

في حديثنا السابق عن مصفوفات الدقة، فيجب عليك الآن أن تكون قد تعرفت على المصطلحات $TP + FN$ وهي مجموع القيم في الصف الأول، وهي تساوي المجموع الحقيقي للبيانات التي تنتمي للتصنيف الإيجابي Positive Class. استخدام مصفوفات الدقة يسهل علينا مقارنة الحساسية في النماذج:

- `ham_only`: $\frac{0}{0+42} = 0$
- `spam_only`: $\frac{42}{42+0} = 1$
- `words_list_model`: $\frac{18}{18+24} \approx .429$

بما أن النموذج `ham_only` لم يكون فيه أي قيمه صحيحة في العمود True Positive، فحصل على أسوأ النتائج 0 في الحساسية. على الجانب الآخر، النموذج `spam_only` كانت الدقة فيه قليلة جداً إلا انه حسب على أفضل النتائج في الحساسية 1 لأنه توقع جميع رسائل البريد المزعجة بشكل صحيح. النتيجة المتدنية للنموذج `words_list_model` تعني أنه عادة ما يفشل في توقع رسائل البريد الإلكتروني كمزعجة؛ ولكن على العكس، النموذج أفضل بكثير من النموذج `ham_only`.

النوعية

تسمى **النوعية Specificity** أيضاً بـ **معدل التوقع والنتيجة السلبيان True Negative Rate** يقيس نسبة البيانات التي تنتمي إلى التصنيف السلبي وتوقعها النموذج بشكل صحيح: 📝

$$\text{Specificity} = \frac{TN}{TN + FP}$$

المصطلح $TN + FP$ يساوي المجموع الحقيقي للبيانات التي تنتمي للتصنيف السلبي Negative Class. مرة أخرى، استخدام مصفوفات الدقة يسهل علينا مقارنة النوعية في النماذج:

- `ham_only`: $\frac{758}{758+0} = 1$
- `spam_only`: $\frac{0}{0+758} = 0$
- `words_list_model`: $\frac{752}{752+6} \approx .992$

كما في الحساسية، النتائج من 0 إلى 1 من الأسوأ إلى الأفضل. لاحظ أن `ham_only` حصل على أفضل النتائج في النوعية و أسوأ النتائج في الحساسية، بينما `spam_only` حصل على أسوأ النتائج في النوعية والأفضل في الحساسية. بما أن هذه النماذج تتوقع نتيجة واحدة فقط، فأنها تصنف جميع القيم من التصنيف الآخر بشكل خاطئ، يظهر ذلك في القيم العالية للنوعية والحساسية. الفرق في النتائج في النموذج `words_list_model` أقل.

على الرغم أن الحساسية والتصنيف تعرفنا على خصائص مختلفة في النموذج، نحتاج أن نربط تلك النتائج في أداتا التقييم مع بعضها البعض باستخدام حد فصل التصنيف

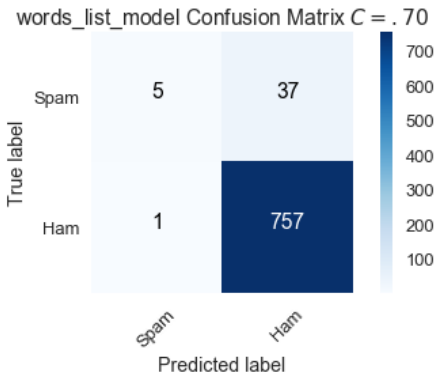
حد فصل التصنيف

قيمة **حد فصل التصنيف Classification Threshold** تحدد إي تصنيف تنتمي إليه قيمة ما؛ القيم على الجانبين المختلفين من حد الفصل يتم يختلف تصنيفها عن بعضها. لنتذكر أن النتائج في الانحدار اللوجستي عبارة عن احتماليات إذا كانت النقطة تنتمي إلى التصنيف الإيجابي. إذا كانت الاحتمالية أكبر من حد الفصل، فأنها تنتمي إلى التصنيف الإيجابي، إذا كانت أقل من حد الفصل، فأنها تنتمي إلى التصنيف السلبي. في مثالنا، لنجعل $f_{\hat{\theta}}$ نموذجنا اللوجستي و C هو حد الفصل. إذا كانت $f_{\hat{\theta}}(x) > C$، فإن x تصنف رسالة بريد مزعج؛ إذا كانت $f_{\hat{\theta}}(x) < C$، فإن x تصنف كرسالة غير مزعجة. تفصل مكتبة `scikit-learn` عند تعادل النتيجة مع حد الفصل للتصنيف السلبي، إذاً عندما تكون $f_{\hat{\theta}}(x) = C$، فإن x يتم تصنيفها كرسالة غير مزعجة.

يمكننا تقييم أداء النموذج مع حد الفصل C باستخدام مصفوفة الدقة. مصفوفة الدقة للنموذج `words_list_model` التي سبق أن عرضناها تستخدم القيمة الافتراضية لحد الفصل في مكتبة `scikit learn` وهي $C = .50$.

الرفع من حد الفصل إلى $C = .70$، يعني أننا نصنف رسالة البريد الإلكتروني x كمزعجة عندما تكون الاحتمالية $f_{\hat{\theta}}(x)$ أعلى من .70، النتائج لمصفوفة الدقة عندما يكون حد الفصل .70 كالتالي:

```
# إيجاد احتماليات التوقع
words_list_prediction_probabilities = words_list_model.predict_proba(X_train)[:, 1]

# عندما تكون الاحتمالية أكبر من .70 محدد رسالة البريد كرسالة مزعجة، والعكس غير ذلك
words_list_predictions = [1 if pred >= .70 else 0 for pred in words_list_prediction_probabilities]

# إنشاء مصفوفة الدقة
high_classification_threshold = confusion_matrix(y_train, words_list_predictions, labels=[1, 0])

# رسم مصفوفة الدقة
plot_confusion_matrix(high_classification_threshold, classes=class_names,
                      title='words_list_model Confusion Matrix $C = .70$')
```



عند رفع حد الفصل لتحديد ما إذا كانت رسالة البريد الإلكتروني مزعجة أم لا، يوجد 13 رسالة بريد مزعجة تم تحديدها بشكل صحيح عندما كانت $C = 0.50$ والآن هي مصنفة بشكل خطأ:

$$\text{Sensitivity } (C = .70) = \frac{5}{42} \approx .119$$

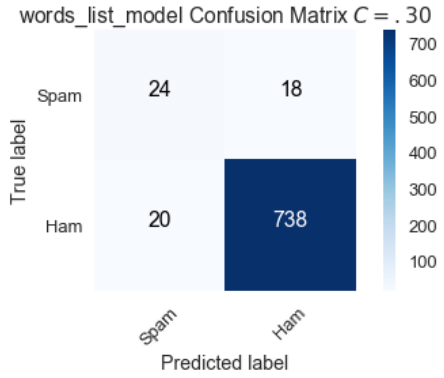
$$\text{Specificity } (C = .70) = \frac{757}{758} \approx .999$$

عند المقارنة بالقيمة الافتراضية، قيمة حد فصل أعلى عند $C = .70$ تزيد من النوعية وتقلل من الحساسية.

التقليل من حد الفصل إلى $C = .30$، يعني أن أي رسالة بريد إلكتروني x تصنف كمزعجة عندما تكون الاحتمالية $f_{\hat{\theta}}(x)$ أكبر من .30، نتيجة مصفوفة الدقة كالتالي:

```
# عندما تكون الاحتمالية أكبر من .30 محدد رسالة البريد كرسالة مزعجة، والعكس غير ذلك
words_list_predictions = [1 if pred >= .30 else 0 for pred in words_list_prediction_probabilities]
```

```
# إنشاء مصفوفة الدقة
low_classification_threshold = confusion_matrix(y_train, words_list_predictions, labels=[1, 0])

# رسم مصفوفة الدقة
plot_confusion_matrix(low_classification_threshold, classes=class_names,
                      title='words_list_model Confusion Matrix $C = .30$')
```



عند التقليل من حد الفصل لرسائل البريد الإلكتروني المزعجة، تم تصنيف 6 رسائل بريد إلكتروني مزعجة كانت مصنفة بشكل خاطئ عندما كانت $C = .50$. وهي الآن مصنفة بشكل صحيح. ولكن، هناك عدد أكبر من القيم السلبية في الحقيقة وتم توقعها إيجابياً False Positive:

$$\text{Sensitivity } (C = .30) = \frac{24}{42} \approx .571$$

$$\text{Specificity } (C = .30) = \frac{738}{758} \approx .974$$

بالمقارنة مع القيمة الافتراضية، حد فصل أقل عند $C = .30$ زاد من الحساسية وقلل من النوعية.

نقوم بالتعديل من نتائج الحساسية والنوعية للنموذج عن طريق تعديل حد الفصل. على الرغم أننا نرغب في الحصول على أفضل النتائج في الحساسية والنوعية، يمكن أن نرى من مصفوفات الدقة بحدود فصل مختلفة أن هناك مقايضة تحدث. الزيادة في الحساسية تؤدي إلى النقص في النوعية والعكس.

منحنيات ROC

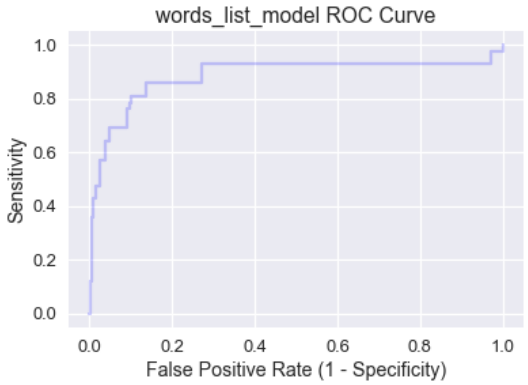
يمكننا حساب الحساسية والنوعية أيًّا كانت قيمة حد الفصل بين 0 و 1 ورسمها. كل قيمة لحد الفصل C مرتبطة بالحساسية والنوعية معًا. **منحنى خصائص تشغيل المُستقْبِل ROC (Receiver Operating Characteristic) Curve** يختلف بعض الشيء؛ بدلاً من رسم النقطة (الحساسية، النوعية)، يرسم (الحساسية، 1-النوعية)، وفيها 1-النوعية معرفة بمعدل القيم السلبية التي تم توقعها إيجابياً False Positive: 📝

$$\text{False Positive Rate } = 1 - \frac{TN}{TN + FP} = \frac{TN + FP - TN}{TN + FP} = \frac{FP}{TN + FP}$$

نقطة في منحنى ROC تمثل الحساسية ومعدل القيم السلبية التي تم توقعها إيجابياً False Positive المرتبطة بقيمة حد الفصل.

منحنى ROC لنموذج `words_list_model` يمكن حسابها باستخدام دالة منحنى ROC في مكتبة `scikit-learn`:

```
from sklearn.metrics import roc_curve

words_list_model_probabilities = words_list_model.predict_proba(X_train)[:, 1]
false_positive_rate_values, sensitivity_values, thresholds = roc_curve(y_train, words_list_model_probabil

plt.step(false_positive_rate_values, sensitivity_values, color='b', alpha=0.2,
         where='post')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('Sensitivity')
plt.title('words_list_model ROC Curve')
```



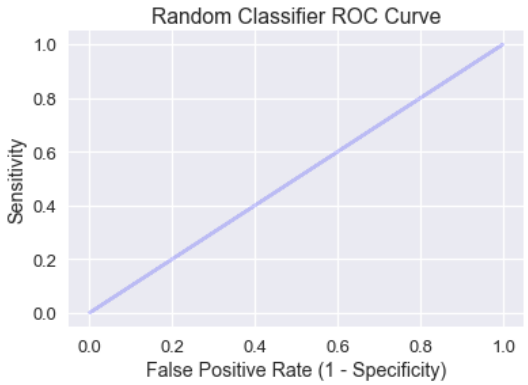
لاحظ أننا كُلما انتقلنا من اليسار إلى اليمين عبر المنحنى، تزيد الحساسية وتقل النوعية. بشكل عام، أفضل حد للفصل تتفاعل عندما تتكون الحساسية والنوعية عالية (قيمة أقل لمعدل القيم السلبية التي تم توقعها إيجابياً False Positive)، لذا النقاط في الأعلى يساراً هي النقاط المثالية.

لننظر بشكل أكبر للزوايا الأربعة في الرسم البياني السابق:

- (0, 0): النوعية $= 1$، يعني أن جميع البيانات في التصنيف السلبي تم تصنيفها بشكل صحيح، ولكن الحساسية $= 0$، إذاً النموذج لا يحتوي على قيم إيجابية وتم توقعها إيجابياً True Positive. حد الفصل عند النقطة (0, 0) فيه $C = 1.0$، وهو مطابق لنتائج النموذج `ham_only` بما أنه لا يمكن لأي رسالة بريد إلكتروني أن تحصل على احتمالية أعلى من 1.0.
- (1, 1): النوعية $= 0$، يعني أن النموذج لا يحتوي توقعات سلبية وتم توقعها سلبياً True Negative، ولكن الحساسية $= 1$، يعني أن جميع البيانات في التصنيف الإيجابي تم تصنيفها بشكل صحيح. الفصل عند النقطة (1, 1) فيه $C = 0.0$، وهو مطابق لنتائج النموذج `spam_only` بما أنه لا يمكن لأي رسالة بريد إلكتروني أن تحصل على احتمالية أعلى من 0.0.
- (0, 1): كلاهما النوعية والحساسية $= 1$، يعني عدم وجود توقعات إيجابية ونتائج سلبية False Positive أو توقعات سلبية ونتائج إيجابية False Negative. نموذج ذو منحنى ROC يساوي (0, 1) لدية قيمة مثالية لحد فصل C للنموذج.
- (1, 0): كلاهما النوعية والحساسية $= 0$، يعني عدم وجود توقعات إيجابية ونتائج إيجابية True Positive أو توقعات سلبية ونتائج سلبية True Negative. نموذج ذو منحنى ROC يساوي (1, 0) لدية قيمة لحد فصل C يتوقع دائمًا نتائج خاطئة لجميع البيانات.

النموذج الذي يتوقع التصنيف بشكل عشوائي يكون فيه المنحنى ROC خط مستقيم مائل تكون فيه النقاط فيها الحساسية ومعدل القيم السلبية التي تم توقعها إيجابياً False Positive متساوية:

```
plt.step(np.arange(0, 1, 0.001), np.arange(0, 1, 0.001), color='b', alpha=0.2,
         where='post')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('Sensitivity')
plt.title('Random Classifier ROC Curve')
```



النموذج العشوائي الذي يتوقع الاحتمالية p لقيمة مُدخلة x سيتوقع توقعات إيجابية ونتائج إيجابية True Positive أو توقعات إيجابية ونتائج سلبية False Positive لـ p، لذا الحساسية و ومعدل التوقعات الإيجابية والنتائج السلبية False Positive متساويان.

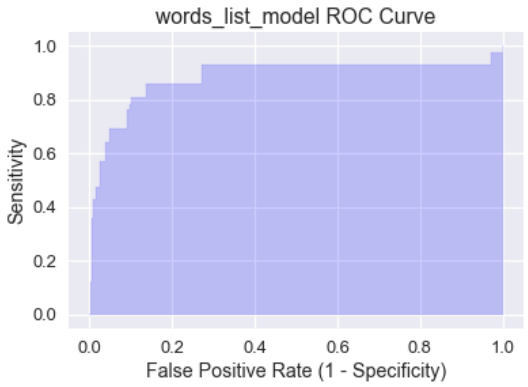
نريد أن يكون منحنى ROC في النموذج أعلى من خط النموذج العشوائي، ونصل الآن لمفهوم AUC.

AUC

المساحة أسفل المنحنى (Area Under Curve (AUC هي المساحة أسفل منحنى ROC وتعمل كأداة لتلخيص أداء النموذج. المساحة أسل المنحنى للنموذج `words_list_model` تم رسمها في الأسفل ويمكن حسابها باستخدام دالة AUC في مكتبة `scikit-learn`:

```
plt.fill_between(false_positive_rate_values, sensitivity_values, step='post', alpha=0.2,
                 color='b')

plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('Sensitivity')
```

```
plt.title('words_list_model ROC Curve')
```




```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_train, words_list_model_probabilities)
```

```
0.9057984671441136
```

يمكن تفسير AUC على أنها احتمالية أن يقوم النموذج بتعيين قيمة عالية للاحتمالية لنقطة مختارة بشكل عشوائي تنتمي للتصنيف الإيجابي من نقطة مختارة بشكل عشوائي تنتمي للتصنيف السلبي. القيمة المثالية لـ AUC هي 1 يكون فيها النموذج مثالي (منحنى ROC سيكون (0, 1)). حصول النموذج `words_list_model` على مساحة أسفل المنحنى AUC تساوي 0.906 يعني أن النموذج من المحتمل أن يتوقع الرسائل المزعجة كمزعجة بنسبة 90.6% أكثر من توقعه الرسائل السليمة كمزعجة.

عند التحقق من ذلك، المساحة أسفل المنحنى AUC للنموذج العشوائي تساوي 0.5، على الرغم أن ذلك أيضاً قابل للتغير بشكل قليل بناءًا على العشوائية. النموذج الفعال والمفيد تكون فيه AUC أعلى بكثير من 0.5، والتي حققها النموذج `words_list_model`. إذا كانت المساحة أسفل المنحنى أقل من 0.5، فإن النموذج أداءه أسوأ من أداء التوقعات العشوائية.

ملخص تقييم النموذج اللوجستي

المساحة أسفل المنحنى AUC هي أداة لتقيم النماذج في البيانات ذات التصنيف الغير متوازن. بعد تدريب النموذج، من الممارسات الممتازة هي إيجاد منحنى ROC وحساب AUC لتحديد الخطوة التالية. إذا كان AUC عالي، نستخدم منحنى ROC لتحديد أفضل قيمة لحد الفصل. ولكن، إذا كانت قيمة AUC غير مرضية، فمن الأفضل القيام بالمزيد من التحليل الاستكشافي للبيانات و إختيار قيم أفضل للخصائص للتحسين من أداء النموذج.

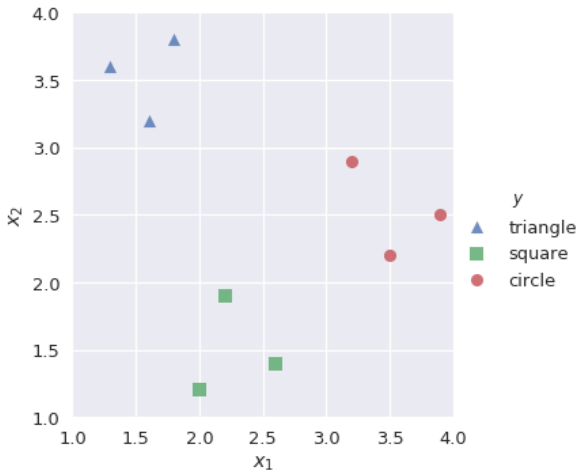
التصنيف متعدد التصنيفات

حتى الآن يقوم النموذج بتصنيف ثنائي فيها النتائج واحد من نتيجتين؛ مثلاً، صنفنا الرسائل كمزعجة أو غير مزعجة. ولكن، الكثير من مشاكل في علم البيانات تحتوي على **تصنيفات مُتعددة التصنيفات Multiclass Classification**، فيه نريد أن نصنف نتائج البريد الإلكتروني إلى واحد من عدة تصنيفات. مثلاً، تحديد رسائل البريد ما إذا كانت عائلية، من الأصدقاء، خاص بالعمل أو الإعلانات والعروض. لحل هذا النوع من المشاكل، نستخدم طريقة جديدة يطلق عليها **تصنيف واحد-ضد-البقية One-vs-Rest (OvR) classification**. 📝

تصنيف واحد-ضد-البقية

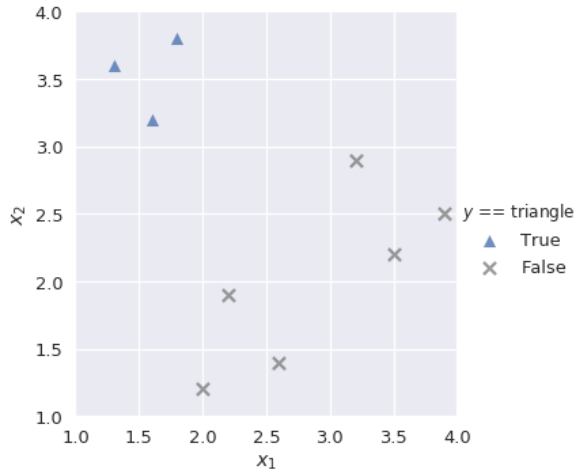
في هذا النوع من التصنيف (يطلق عليه أيضاً واحد-ضد-الجميع One-vs-All أو OvA)، نقوم بتقسيم مشكلة التصنيف المتعدد التصنيفات إلى مشكلة من عدة تصنيفات ثنائية النتائج. مثلاً، يمكن أن تظهر لنا نتائج بيانات التدريب كالتالي:

```
shapes = pd.DataFrame(
    [[1.3, 3.6, 'triangle'], [1.6, 3.2, 'triangle'], [1.8, 3.8, 'triangle'],
     [2.0, 1.2, 'square'], [2.2, 1.9, 'square'], [2.6, 1.4, 'square'],
     [3.2, 2.9, 'circle'], [3.5, 2.2, 'circle'], [3.9, 2.5, 'circle']],
    columns=['$x_1$', '$x_2$', '$y$']
)

sns.lmplot('$x_1$', '$x_2$', data=shapes, hue='$y$', markers=['^', 's', 'o'], fit_reg=False)
plt.xlim(1.0, 4.0)
plt.ylim(1.0, 4.0);
```



هدفنا هو رسم نموذج متعدد التصنيف يقوم بتوقع النتائج من ثلاث `triange`، `square` أو `circle` عند تمرير القيم x_1 و x_2 إليه. أولاً نقوم ببناء نموذج ثنائي بأسم `lr_triangle` يقوم بتوقع النتائج ما إذا كانت مُثلثات `triangle` أم لا:

```
plot_binary(shapes, 'triangle')
```

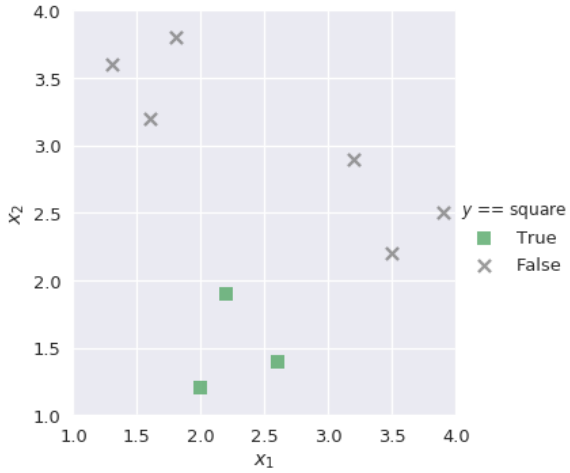


إستعان الكاتب بدالة عرفها بأسم `plot_binary` لمساعدته على إظهار النتائج في رسم بياني، وعرفها كالتالي:

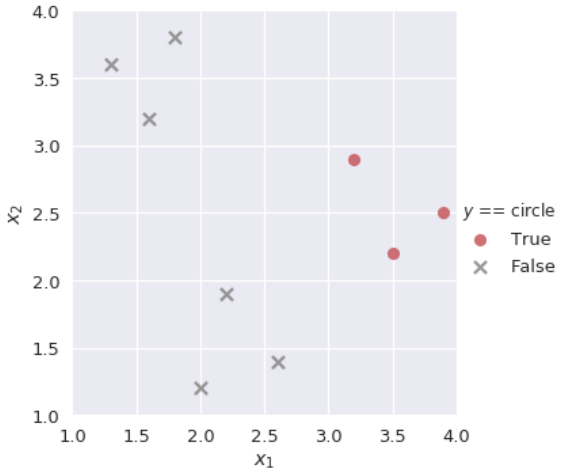
```


markers = {'triangle':['^', sns.color_palette()[0]],
           'square':['s', sns.color_palette()[1]],
           'circle':['o', sns.color_palette()[2]]}

def plot_binary(data, label):
    data_copy = data.copy()
    data_copy['$y$ == ' + label] = (data_copy['$y$'] == label).astype('category')

    sns.lmplot('$x_1$', '$x_2$', data=data_copy, hue='$y$ == ' + label, hue_order=[True, False],
               markers=[markers[label][0], 'x'], palette=[markers[label][1], 'gray'],
               fit_reg=False)
    plt.xlim(1.0, 4.0)
    plt.ylim(1.0, 4.0);
```

بنفس الطريقة، نقوم ببناء نموذج `lr_square` و `lr_circle` للتصنيفات الأخرى:

```
plot_binary(shapes, 'square')
```




```
plot_binary(shapes, 'circle')
```



نعلم أن النتائج من الدالة السينية في الانحدار اللوجستي هي عبارة عن احتمالية قيمة بين 0 و 1. لحل التصنيف متعدد التصنيفات لدينا، نبحث عن احتمالية الجانب الإيجابي في كل نموذج ونختار التصنيف الأعلى إيجابية. مثلاً، لو إذا إطلعنا على القيم التالية:

x_1	x_2
3.2	2.5

سيقوم النموذج بإدخال هذه القيم إلى كل النماذج السابقة `lr_square`، `lr_triangle`، و `lr_circle`. نقوم بإيجاد القيمة الاحتمالية الإيجابية لكل من النماذج الثلاثة:

```
from sklearn.linear_model import LogisticRegression

lr_triangle = LogisticRegression(random_state=42)
lr_triangle.fit(shapes[['$x_1$', '$x_2$']], shapes['$y$'] == 'triangle')
proba_triangle = lr_triangle.predict_proba([[3.2, 2.5]])[0][1]

lr_square = LogisticRegression(random_state=42)
lr_square.fit(shapes[['$x_1$', '$x_2$']], shapes['$y$'] == 'square')
proba_square = lr_square.predict_proba([[3.2, 2.5]])[0][1]

lr_circle = LogisticRegression(random_state=42)
lr_circle.fit(shapes[['$x_1$', '$x_2$']], shapes['$y$'] == 'circle')
proba_circle = lr_circle.predict_proba([[3.2, 2.5]])[0][1]
```

lr_triangle	lr_square	lr_circle
0.145748	0.285079	0.497612

بما أن الاحتمالية الإيجابية الأعلى هي لدى `lr_circle`، فإن النموذج متعدد التصنيفات سيتوقع أن القيمة التي اطلع عليها تنتمي إلى التصنيف `circle`.

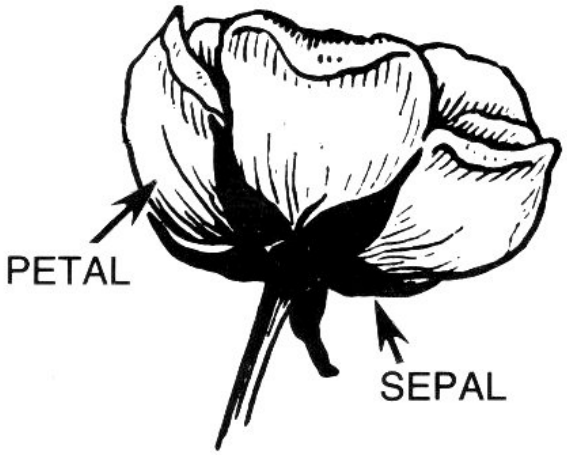
تطبيق عملي: بيانات Iris

تعتبر بيانات Iris إحدى أشهر البيانات المستخدمة في علم البيانات لتعلم مفاهيم تعلم الآلة. هناك ثلاث تصنيفات، كل واحدة تمثل أحد أنواع أزهار السوسن Iris:

- Iris-setosa
- Iris-versicolor
- Iris-virginica

هناك أربع خصائص متوفرة في البيانات:

- Sepal length (cm) طول كأس الزهرة (بالسنتيمتر).
- Sepal width (cm) عرض كأس الزهرة (بالسنتيمتر).
- Petal length (cm) طول بتلة الزهرة (بالسنتيمتر).
- Petal width (cm) عرض بتلة الزهرة (سنتيمتر).



سنقوم بإنشاء نموذج متعدد التصنيفات يقوم بتوقع نوع الزهرة بناءًا على الخصائص الأربعة. أولاً، نقوم بقراءة البيانات:

```
iris = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data',
                   header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'spec

iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
...	...	...	...	...	...
147	6.5	3	5.2	2	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3	5.1	1.8	Iris-virginica

```
150 rows × 5 columns
```

```
X, y = iris.drop('species', axis=1), iris['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=42)
```

بعد تقسيم البيانات إلى بيانات تدريب و اختبار، نقوم بضبط النموذج متعدد التصنيفات على بيانات التدريب. بشكل تلقائي، يقوم نموذج الانحدار اللوجستي `LogisticRegression` في مكتبة `scikit-learn` بتحديد قيمة المتغير `'multi_class='ovr`، والتي تقوم بإنشاء نموذج ثنائي لكل تصنيف مختلف:

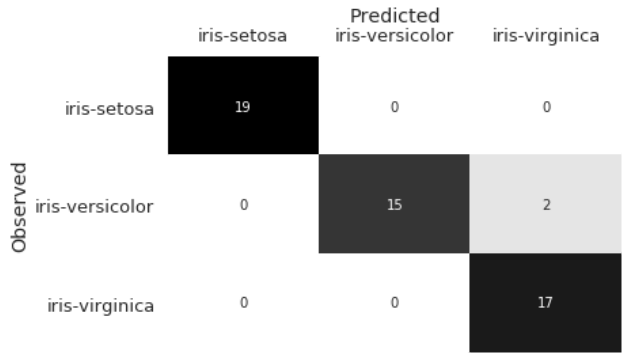
```
lr = LogisticRegression(random_state=42)
lr.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
```

```
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

نقوم بالتوقع باستخدام بيانات الاختبار، ثم نستخدم مصفوفة الدقة لتقييم النتائج:

```
y_pred = lr.predict(X_test)
plot_confusion_matrix(y_test, y_pred)
```

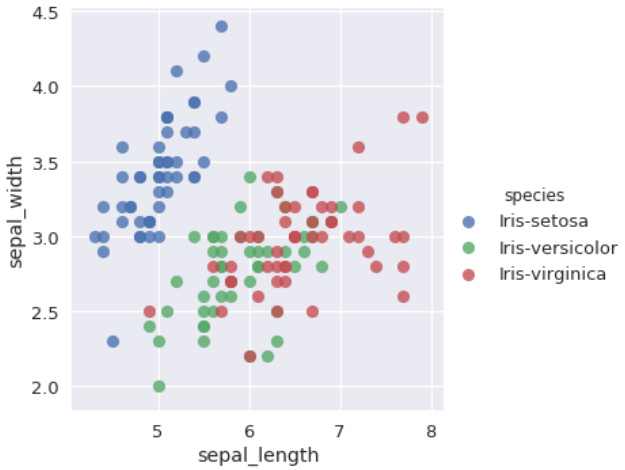


إستخدم الكاتب الدالة `plot_confusion_matrix` والتي عرفها لمساعدته لعرض وتقييم النتائج على شكل مصفوفة الدقة:

```
def plot_confusion_matrix(y_test, y_pred):
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cbar=False, cmap=matplotlib.cm.get_cmap
    plt.ylabel('Observed')
    plt.xlabel('Predicted')
    plt.xticks([0.5, 1.5, 2.5], ['iris-setosa', 'iris-versicolor', 'iris-virginica'])
    plt.yticks([0.5, 1.5, 2.5], ['iris-setosa', 'iris-versicolor', 'iris-virginica'], rotation='horiz
    ax = plt.gca()
    ax.xaxis.set_ticks_position('top')
    ax.xaxis.set_label_position('top')
```

يظهر من مصفوفة الدقة السابقة أن النموذج أخطأ في التوقع مرتين للتصنيف `Iris-versicolor` وتوقعها كتصنيف `Iris-virginica`. عند التحقق من الخصائص `sepal_length` و `sepal_width`، نستطيع أن نرى سبب حدوث ذلك:

```
sns.lmplot(x='sepal_length', y='sepal_width', data=iris, hue='species', fit_reg=False);
```



تتداخل الكثير من النقاط في للتصنيفات `Iris-versicolor` و `Iris-virginica`. على الرغم أن الخصائص الأخرى (`petal_width` و `petal_length`) تقدم معلومات أكثر للتفريق بين التصنيفان، إلا أن النموذج أخطأ في تصنيف نوعين من الزهور.

كما في العالم الحقيقي، تحدث التصنيفات الخطأ عندما يكون تصنيفان متشابهان في الخصائص. مصفوفات الدقة مفيدة كونها تطلعنا على الأخطاء التي يقع فيها النموذج، أيضاً تقدم لنا أفكار عن أي نوع من الخصائص نحتاج أن نوجدها من أجل تحسين أداء النموذج.

التصنيف متعدد النتائج

نوع آخر من مشاكل التصنيف هو **التصنيف متعدد النتائج Multilabel Classification**، وفيه يكون للنتيجة أكثر من تصنيف. مثال لذلك في نظام لتصنيف الملفات: الملف قد يكون ذو إيجابي أو سلبي، ذو محتوى ديني أو غير ديني، و متفتح أو محافظ. مشكلة متعددة النتائج قد تكون أيضاً متعددة التصنيفات؛ قد نرغب أن يكون نظام التصنيف للملفات لدينا يفرق بين قائمة من الأنواع، أو يحدد اللغة التي كُتب فيها الملف.

يمكن أن نطبق التصنيف متعدد النتائج ببساطة عبر تدريب نماذج منفصلة لكل نوع من النتائج. لتصنيف نقطة ما، نقوم بجمع جميع نتائج توقعات النماذج.

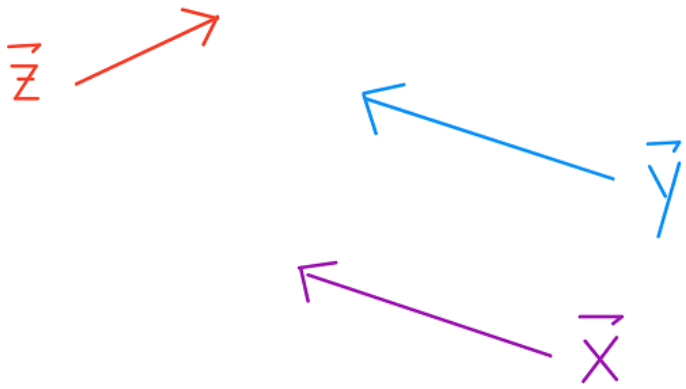
ملخص التصنيف متعدد الاحتمالات

مشاكل التصنيف قد تكون صعبة. في بعض الأحيان، نحتاج في المشكلة للتفريق بين أكثر من تصنيف؛ وفي بعض الأحيان قد نريد تعين أكثر من نتيجة لكل قيمة يطلع علينا النموذج. نستعين بما نعرفه عن النماذج ثنائية النتائج لبناء أنظمة تصنيف متعدد التصنيفات والنتائج والتي قادرة على أداء تلك المهام.

مراجعة فضاء المتّجهات

تعريف المتّجه

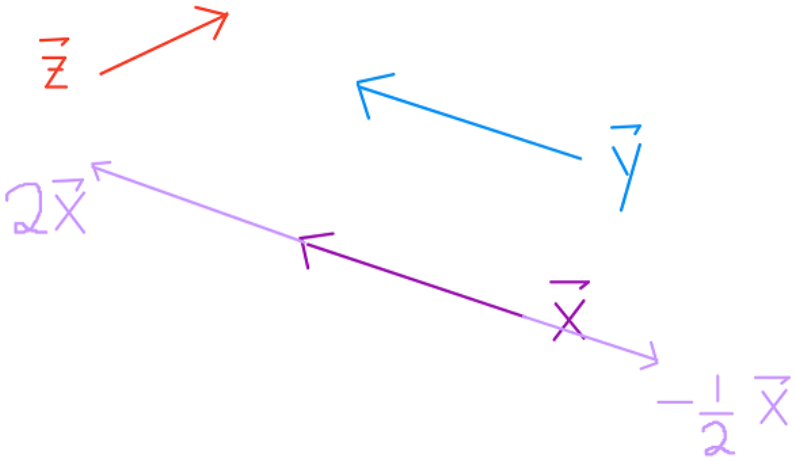
تعرف المتّجه بطولها و اتجاهها:



لاحظ أن المتّجه $\vec{x}$ و $\vec{y}$ بنفس الطول والاتجاه. لذا هما متجهتان متساويتان.

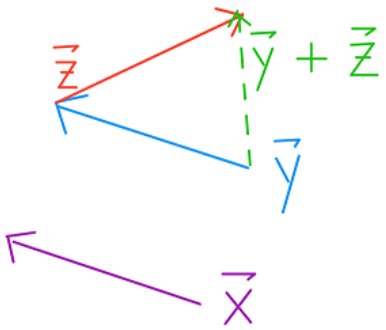
زيادة المتّجهات

الزيادة في المتّجه يعني التعديل في طولها:



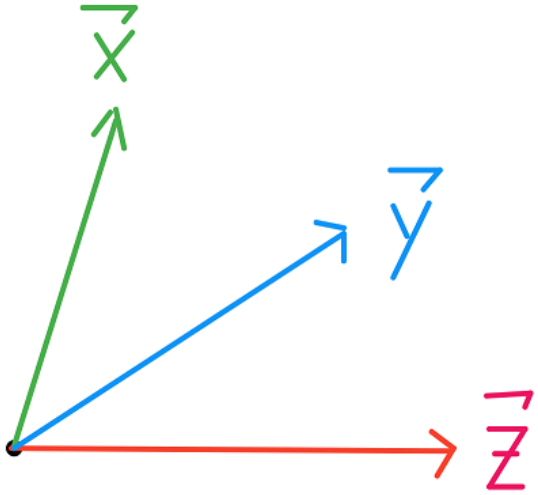
لاحظ أن $2\vec{x}$ و $\vec{y}$ لديهما نفس الاتجاه ولكن بأطوال مختلفة. لذا هما غير متساويان.

لجمع متّجهتان $\vec{y} + \vec{z}$، نأخذ خطوة بحسب طول $\vec{y}$، ثم مباشرة نأخذ خطوة بحسب طول $\vec{z}$ (أو العكس). يطلق على هذه الخطوات طريقة المثلث، فيها تربط بداية المتّجه الأولى بنهاية المتّجه الثانية.



رموز المتّجهات

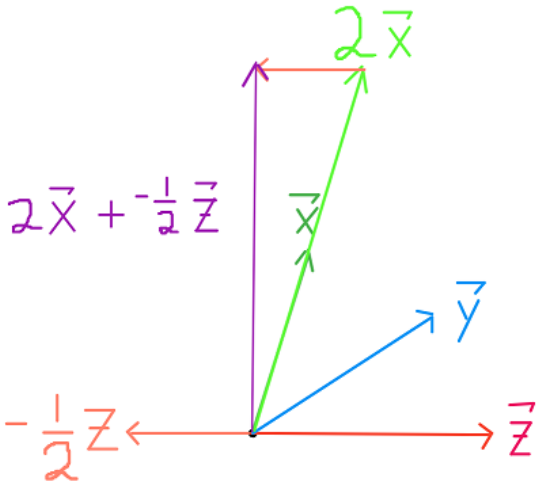
يتم تمثيل المتّجهات عادة بالإحداثيات الديكارتية Cartesian Coordinates:



$$\vec{x} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad \vec{z} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

باستخدام هذه الرموز، فإن حساب العملية الرياضية السابقة يصبح سهلاً:

$$2\vec{x} = \begin{bmatrix} 2 \\ 8 \end{bmatrix}, \quad -\vec{0}.5z = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \quad 2x + \vec{-}0.5z = \begin{bmatrix} 0 \\ 8 \end{bmatrix}$$



يمكن الزيادة والإضافة في المتّجهات على شكل أجزاء:

$$a\vec{x} + b\vec{y} = \begin{bmatrix} a\,x_1 & + & b\,y_1 \\ & \vdots & \\ a\,x_n & + & b\,y_n \end{bmatrix}$$

المتّجه 1

في أي مساحة d للفضاء أو الأبعاد، $\vec{1}$ هي متّجهة تحتوي على الرقم 1 مكرراً: $\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$

مدى مجموعة من المتّجهات

مدى مجموعة من المتّجهات $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_p\}$ هي جميع التركيبات الخطية المحتملة. لهذه المتّجهات p:

$$\{c_1\,\vec{v}_1 + c_2\,\vec{v}_2 + \cdots + c_p\,\vec{v}_p \;:\; \forall c_i \in F\}$$

فيها F هي مجال فضاء المتّجه.

فضاء المتّجه

فضاء المتّجه V هو مدى مجموعة من المتّجهات $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_p\}$، فيها كل $\vec{v}_i$ عبارة عن عمود متّجه بطول $n \times 1$.

الفضاءات الفرعية للمتّجه

الفضاء الفرعي U لـ V هو مدى مجموعة من المتّجهات $\{\vec{u}_1, \ldots, \vec{u}_q\}$ فيها كل $\vec{u}_i \in V$. يعني ذلك أن كل متّجه في U هي أيضاً في V.

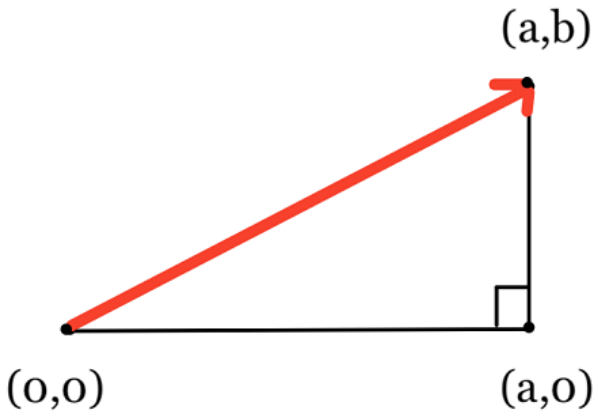
الزوايا بين المتّجهات

عند الربط بين نهاية متّجهتان معاً دون التعديل في اتجاهها، يمكن قياس الزاوية بينهما:

طول المتّجه

الفكرة في $\mathbb{R}^2$:

نتذكر طريقة المثلث لجمع متّجهتان. إذا قمنا بإضافة متّجهتان عاموديتان $\vec{a} + \vec{b}$ في $\mathbb{R}^2$، فإننا نعلم أن المتّجه الناتجة هي الوتر. في هذه الحالة، نعلم أيضاً أن طول $\vec{a} + \vec{b}$ سيكون مطابقاً لنظرية فيثاغورس: $\sqrt{a^2 + b^2}$:



المعادلة العامة لطول $\vec{v} \in \mathbb{R}^n$:

$$\begin{aligned} ||\vec{v}|| &= \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} \\ &= \sqrt{\vec{v} \cdot \vec{v}} \end{aligned}$$

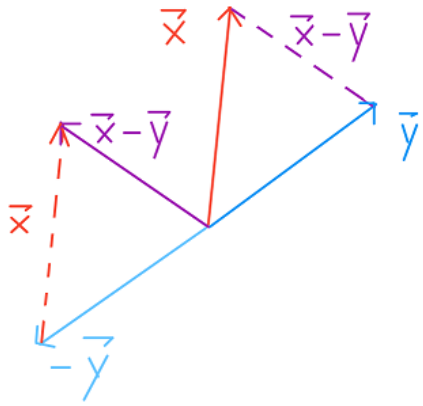
فيه العامل الأخير هو حاصل عملية الضرب:

$$\begin{aligned} \vec{x} \cdot \vec{y} &= x_1\, y_1 + x_2\, y_2 + \cdots + x_n\, y_n \\ &= ||x||\, ||y||\, \cos\theta \end{aligned}$$

التعبير الأول معروف أيضاً بأسم التعريف الجربي للجداء النقطي، والثاني يعرف بالتعريف الهندسي. لاحظ أن نتيجة الضرب هي حاصل الضرب الداخلي المعرف في المتجهات في $\mathbb{R}^n$

المسافة بين متّجهتان

$$dist(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||$$

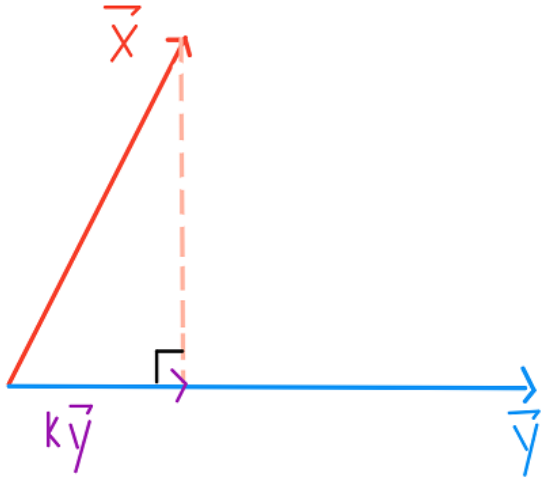


المتّجهات المتعامدة

لمتّجهتان غير صفرية لتصبحان متعامدة، يجب أن يطابقا شرط أن $\vec{x} \cdot \vec{y} = 0$. بما أن طولهما لا يساوي صفر، الطريقة الوحيدة لتصبحا المتّجهتان عاموديتان عندما تكون $\cos\theta = 0$. عندما تكون θ تساوي 90 درجة، يكون لدينا ما نعرفه بالزاوية القائمة.

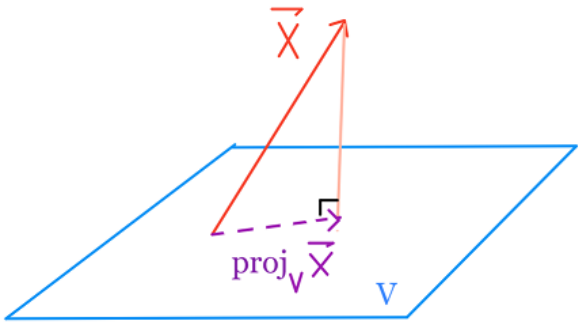
إسقاط المتّجهات

لإسقاط متّجهة $\vec{x}$ على متّجه أخرى $\vec{y}$، فنحن نريد إيجاد $k\vec{y}$ الأقرب إلى $\vec{x}$:



بناءًا على نظرية فيثاغورس، نعلم أن k يجب أن تكون العدد الذي فيه $\vec{x} - k\,\vec{y}$ عموديه إلى $\vec{y}$، إذاً $k\vec{y}$ هي الإسقاط (المتعامد) لـ $\vec{x}$ على $\vec{y}$.

بنفس الطريقة، لإسقاط متّجه $\vec{x}$ على أي فضاء المتجهات ممتد بواسطة أي مجموعه من المتّجهات $\{\vec{v_1}, \vec{v_2}, \ldots, \vec{v_p}\}$، فلا نزال نحتاج لإيجاد التركيبات الخطية المحتملة $\{k_1\,\vec{v_1} + k_2\,\vec{v_2} + \cdots + k_p\,\vec{v_p}\}$ الأقرب إلى $\vec{x}$:



مراجعة الاستدلال الإحصائي

مقدمة

هذا الملحق يستعرض الاستدلال باستخدام اختبارات التبادل والتمهيد. تم التحدث عن هذه المواضيع في الفصل 12 و 13 من كتاب داتا 8.

على الرغم أن علماء البيانات يعملون على عينات من البيانات، إلا أننا دائماً مهتمين بالتعميم على المجمع الإحصائي التي أتت منة البيانات. هذا الملحق يتحدث عن طرق إجراء *الاستدلال الإحصائي Statistical Inference*، طريقة لبناء استنتاجات عن كل المجتمع الإحصائي باستخدام البيانات.

نعتمد في الاستدلال الإحصائي بشكل عام على طريقتين: اختبار الفرضيات ومجالات الثقة. في الماضي القريب، اعتمدت هذه الأساليب بشكل كبير على النظرية العادية، والتي هي فرع من الإحصائيات التي تتطلب افتراضيات أساسية عن المجتمع الإحصائي. اليوم، التطور السريع والذي وفر حوسبة قوية أمكن من إيجاد طرق جديدة من الأساليب والتي تعتمد على *إعادة التشكيل Resampling* والتي تعمم على أنواع كثيره من المجتمعات الإحصائية.

اختبار الفرضيات و مجال الثقة

سيكون هذا الملحق مراجعة بسيطة لاختبار الفرضيات باستخدام التمهيد واختبارات التبادل. نتوقع معرفة بهذه العناوين كونها تم التحدث بشكل كبير عنها في كتاب التفكير الحسابي والاستنتاجي، الكتاب الخاص بمادة داتا 8. للحصول على معلومات وشرح أكثر عن المواضيع التي سيتم شرحها هنا، أطلع على الفصل 11، 12 و 13 من كتاب التفكير الحسابي والاستنتاجي.

اختبارات الفرضية

عند تطبيق مفاهيم علم البيانات في مختلف المجالات، عادة ما نواجه أسئلة عامة. مثلاً، هل شرب القهوة يسبب حرمان النوم؟ هل تتسبب السيارات ذاتية القيادة بحوادث أكثر من السيارات غير ذاتية القيادة؟ هل الدواء x يساعد على علاج الالتهابات الرئوية؟ للمساعدة في الإجابة على هذه الأسئلة، نستخدم اختبارات الفرضية لإيجاد نتائج علمية بناءًا على الأدلة والبيانات التي اطلعنا عليها.

بما أن عملية جمع البيانات عادة ما تحتوي على خطوات غير دقيقة، عادة ما نكون غير متأكدين من الانماط الموجودة في تلك البيانات إن كانت بسبب التشويش أو بسبب بيانات حقيقة عن الالتهاب الرئوي. تساعدنا اختبارات الفرضية على تحديد ما إذا كان النمط في البيانات حدث بسبب التقلبات العشوائية في طريقة جمعنا للبيانات.

سنبدأ بمثال لاستكشاف اختبارات الفرضية. الجدول baby يحتوي على بيانات أوزان الأطفال وقت الولادة. تم تسجيل وزن المواليد بالأونصة وما إذا كانت الأم تدخن أثناء حملها:

لتحميل البيانات baby.csv اضغط هنا.

```
import pandas as pd
import numpy as np

baby = pd.read_csv('baby.csv')
baby = baby.loc[:, ["Birth Weight", "Maternal Smoker"]]

baby
```

	Birth Weight	Maternal Smoker
0	120	FALSE
1	113	FALSE
2	128	TRUE
...	...	...
1171	130	TRUE
1172	125	FALSE
1173	117	FALSE

```
1174 rows × 2 columns
```

التصميم

نريد أن نرى ما إذا كان تدخين الأم له علاقة بوزن المولود. لإعداد الاختبار لفرضيتنا، يمكننا تمثيل وجهتي النظر العامة كالتالي:

- **فرضية العدم Null hypothesis**: في المجتمع الإحصائي، يكون التوزيع للبيانات لأوزان المواليد للأمهات غير المدخنات مساوي للتوزيع لأوزان المواليد من الأمهات المدخنات. الفرق الذي يحدث في البيانات كان بالصدفة.

- **الفرضية البديلة Alternative hypothesis**: في المجتمع الإحصائي، متوسط الوزن أثناء الولادة للمواليد من الأمهات المدخنات كان أقل من المواليد من أمهات غير مدخنات.

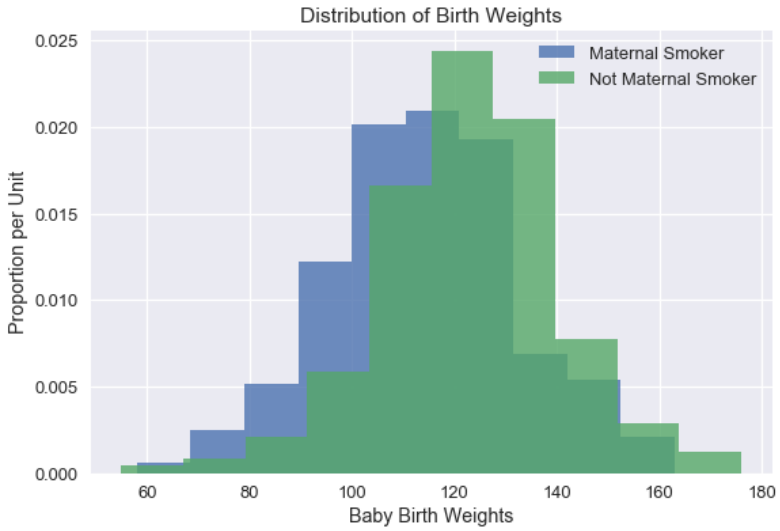
هدفنا الرئيسي هو اتخاذ قرار بين هذه الفرضيات. نقطة مهمة يجب الانتباه لها وهي أننا أوجدنا هذه الفرضيات بناءًا على *المتغيرات Parameters* من البيانات وليس من تجاربنا. مثلاً، يجب ألا نقوم ببناء فرضية العدم مثل "وزن مواليد الأمهات المدخنات سيكون مساوي لوزن مواليد الأمهات غير المدخنات"، نظرًا لوجود تباين بشكل طبيعي في النتائج.

تؤكد فرضية العدم أنه إذا كانت النتائج تظهر عكس ما توقعته الفرضية، فإن ذلك الفرق كان بالصدفة. وبشكل غير رسمي، تقول الفرضية البديلة إن الفرق والاختلاف التي أطلعت عليه "حقيقي".

يجب أن نلقي نظرة أكثر على شكل فرضيتنا البديلة. في الوضع الحالي، لاحظ أننا سنرفض فرضية العدم إذا كان وزن المواليد من الأمهات المدخنات أقل بشكل واضح وكبير من أوزان المواليد من الأمهات غير المدخنات. بمعنى آخر، الفرضية البديلة تشمل / تدعم جانب واحد من التوزيع. نطلق على ذلك **الفرضية البديلة من جانب واحد One-sided Alternative Hypothesis**. بشكل عام، نستخدم هذا النوع من الفرضيات البديلة إذا كان لدينا سبب مقنع بأنه لا يمكن أن نرى متوسط وزن لمواليد من أمهات مدخنات أعلى.

لرسم البيانات، قمنا بإنشاء مدرج تكراري لوزن المواليد من الأمهات المدخنات وغير المدخنات:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(9, 6))
smokers_hist = (baby.loc[baby["Maternal Smoker"], "Birth Weight"]
                .hist(density=True, alpha=0.8, label="Maternal Smoker"))
non_smokers_hist = (baby.loc[~baby["Maternal Smoker"], "Birth Weight"]
                    .hist(density=True, alpha=0.8, label="Not Maternal Smoker"))
smokers_hist.set_xlabel("Baby Birth Weights")
smokers_hist.set_ylabel("Proportion per Unit")
smokers_hist.set_title("Distribution of Birth Weights")
plt.legend()
plt.show()
```



يبدو أن متوسط وزن المواليد من الأمهات المدخنات أقل من وزن المواليد للأمهات غير المدخنات. هل هذا الاختلاف حدث بشكل عشوائي؟ نحاول الإجابة على هذا السؤال باستخدام اختبار الفرضية.

لإجراء اختبار فرضية، نفترض أن نموذج معين يقوم بتوليد البيانات؛ ثم نسأل أنفسنا، ما هي فرصة أن نرى نتائج متطرفة مثل التي رأيناها؟ إذا كانت فرصة مشاهدة تلك النتائج ضئيلة جداً، فقد لا يكون النموذج الذي افترضناه هو النموذج المثالي.

بشكل خاص، نفترض أن فرضية العدم و نموذج إحتماليتها، **نموذج العدم Null Model**، صحيحان. بمعنى آخر، نفترض أن فرضية العدم صحيحة ونركز على القيمة الإحصائية فيها. نموذج الصدفة يقول انه لا يوجد هناك أي فرق؛ توزيع العينات في الرسم البياني مختلفة فقط بمحض الصدفة.

اختبار الإحصائية

في مثالنا، نفترض أن تدخين الأمهات لا يؤثر على وزن المواليد (وأن أي فرق نطلع عليه حدث بالصدفة). من أجل الاختيار بين فرضياتنا، سنستخدم الفرق بين المجموعتين وتكون **اختبار الإحصائية Test Statistic**. اختبار الإحصائية هو:

$$\mu_{\text{smoking}} - \mu_{\text{non-smoking}}$$

إذاً القيم الصغرى (وهي، القيم السلبية الأعلى) لهذه الإحصائية ستفضل الفرضية البديلة. لنقوم بحساب اختبار الإحصائية للبيانات التي أطلعنا عليها:

```
nonsmoker = baby.loc[~baby["Maternal Smoker"], "Birth Weight"]
smoker = baby.loc[baby["Maternal Smoker"], "Birth Weight"]
observed_difference = np.mean(smoker) - np.mean(nonsmoker)
observed_difference
```

```
-9.266142572024918
```

إذا لم يكن هناك أي فرق بين التوزيعين في المجتمع الإحصائي، فإذا كانت الأم مدخنة أو غير مدخنة لن يؤثر على متوسط وزن المولود. بمعنى آخر، القيم `True` و `False` في عمود تدخين الأم لن تأثر على المتوسط.

لذا، من أجل محاكاة اختبار الإحصائية في فرضية العدم، يمكننا إعادة خلط أوزان المواليد بشكل عشوائي بين الأمهات:

```
def shuffle(series):
    '''
    خلط المصفوفة وأعادة تعين الرقم التسلسلي ليتم إضافة القيمة المختلطة لنفس الرقم
    '''
    return series.sample(frac=1, replace=False).reset_index(drop=True)

baby["Shuffled"] = shuffle(baby["Birth Weight"])
baby
```

	Birth Weight	Maternal Smoker	Shuffled
0	120	FALSE	122
1	113	FALSE	167
2	128	TRUE	115
...	...	...	...
1171	130	TRUE	116
1172	125	FALSE	133
1173	117	FALSE	120

```
1174 rows × 3 columns
```

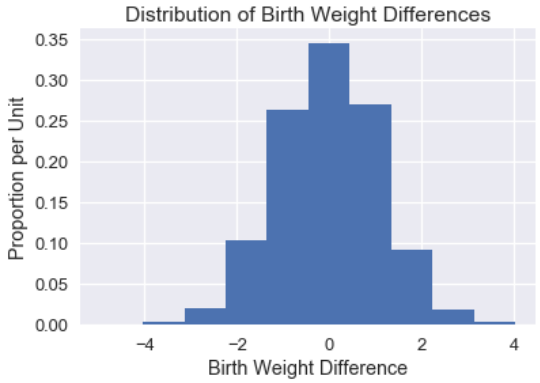
إجراء اختبار التبادل

الاختبارات بناءًا على التبادل العشوائي في البيانات يطلق عليها **اختبارات التبادل Permutation Tests**، في الكود البرمجي التالي، سنحاكي الاختبار الإحصائي أكثر من مرة ونجمع الفرق في مصفوفة:

```
differences = np.array([])

repetitions = 5000
for i in np.arange(repetitions):
    baby["Shuffled"] = shuffle(baby["Birth Weight"])

    # اوجد الفرق بالمتوسط بين مجموعتين عشوائيتين
    nonsmoker = baby.loc[~baby["Maternal Smoker"], "Shuffled"]
    smoker = baby.loc[baby["Maternal Smoker"], "Shuffled"]
    simulated_difference = np.mean(smoker) - np.mean(nonsmoker)

    differences = np.append(differences, simulated_difference)
```

نرسم المدرج التكراري للفرق في المتوسطات

```
differences_df = pd.DataFrame()
differences_df["differences"] = differences
diff_hist = differences_df.loc[:, "differences"].hist(density=True)
diff_hist.set_xlabel("Birth Weight Difference")
diff_hist.set_ylabel("Proportion per Unit")
diff_hist.set_title("Distribution of Birth Weight Differences");
```



من المنطقي أن يتمحور توزيع الفرق في المتوسطات حول الرقم 9 بما أن المجموعتين يجب أن يكون لها نفس المتوسط في فرضية العدم.

من أجل إيجاد استنتاجات لاختبار الإحصائية، يجب علينا حساب القيمة الاحتمالية p-value. القيمة الاحتمالية التجريبية للاختبار هي نسبة الفرق في البيانات التي حاكيناها التي كانت مساوية أو أقل من فرق البيانات الحقيقية:

```
p_value = np.count_nonzero(differences <= observed_difference) / repetitions
p_value
```

```
0.0
```

في بداية اختبار الفرضية عادة ما نختار **قيمة فصل الأهمية Threshold of Significance** (يرمز لها عادة ب α) لقيمة الاحتمالية P-value. إذا كانت قيمة الاحتمالية أقل من قيمة فصل الأهمية، إذا نرفض فرضية العدم. قيم الفصل الأكثر استخداماً هي 0.01 و 0.05، فيها 0.01 تمثل أكثر "تشدد" كوننا نحتاج إلى أدلة أكثر لصالح الفرضية البديلة لرفض فرضية العدم.

في أي من هذه الحالات، نرفض فرضية العدم بما أن قيمة الاحتمالية أقل من قيمة فصل الأهمية.

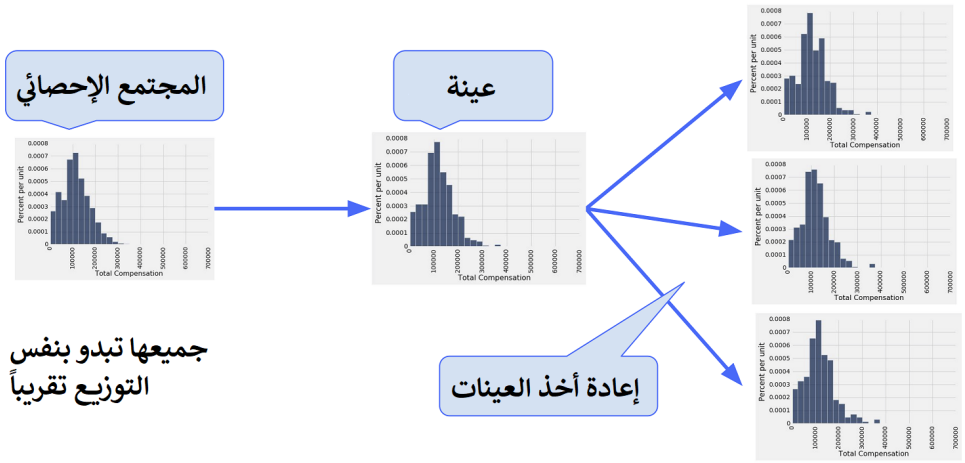
تمهيد فترات الثقة

يحتاج علماء البيانات في كثير من الأحيان لتقدير متغير المجتمع الإحصائي المجهولة باستخدام عينة عشوائية. على الرغم أننا نفضل أخذ عينات كثيرة من المجتمع الإحصائي لإنشاء توزيع للعينات، غالباً ما نكون مقتصرين على عينة واحدة بسبب الوقت والمال.

لحسن الحظ، العينة العشوائية الكبيرة تبدو مشابهة للمجتمع الإحصائي الأصلي. التمهيد Bootstrap يستخدم ذلك لمحاكاة عينات جديدة عن طريق إعادة أخذ العينات من العينة الأصلية.

لتطبيق التمهيد، نقوم بالخطوات التالية:

- أخذ عينة مع بديلة من العينة الأصلية (والتي هي الآن العينة التي تم تمهيدها). هذه العينات يطلق عليها عينات التمهيد. عادة ما نأخذ الآلاف من عينات التمهيد (~10,000).
- حساب الفائدة الإحصائية لكل عينة تمهيدية. يطلق عليها الإحصائية التمهيدية، والتوزيع التجريبي لهذه الإحصائية التمهيدية هي تقريب لتوزيع العينة في الإحصائية التمهيدية.



قد نستخدم توزيع عينات التمهيد لإنشاء فترات الثقة والتي نستخدمها لتوقع قيمة متغير المجتمع الإحصائي.

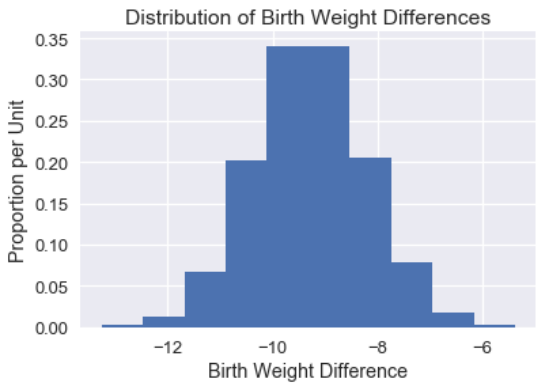
بما أن بيانات الوزن عند الولادة توفر عينة عشوائية حجمها كبير، فقد نعمل على البيانات وكأن الأمهات غير المدخنات يمثلون المجتمع الإحصائي لغير المدخنات. بنفس الطريقة، نقول أن البيانات للأمهات المدخنات يمثلون المجتمع الإحصائي للأمهات المدخنات.

لذا، نعامل العينة الأصلية كتمهيد المجتمع الإحصائي لنتمكن من القيام بخطوات التمهيد:

- نأخذ عينة مع بديل من الأمهات غير المدخنات ونقوم بحساب متوسط وزن المواليد. نأخذ عينة أخرى مع بديل من الأمهات المدخنات ونحسب متوسط وزن المواليد.
- حساب فرق المتوسطات.
- إعادة الخطوات السابقة 10000 مرة، نحصل فيها على 10000 فرق متوسط.

هذه الخطوات تعطينا توزيع تجريبي للعينات للفرق في متوسطات أوزان المواليد:

```
def resample(sample):
    return np.random.choice(sample, size=len(sample))

def bootstrap(sample, stat, replicates):
    return np.array([
        stat(resample(sample)) for _ in range(replicates)
    ])
nonsmoker = baby.loc[~baby["Maternal Smoker"], "Birth Weight"]
smoker = baby.loc[baby["Maternal Smoker"], "Birth Weight"]

nonsmoker_means = bootstrap(nonsmoker, np.mean, 10000)
smoker_means = bootstrap(smoker, np.mean, 10000)

mean_differences = smoker_means - nonsmoker_means

mean_differences_df = pd.DataFrame()
mean_differences_df["differences"] = np.array(mean_differences)
mean_diff = mean_differences_df.loc[:, "differences"].hist(density=True)
mean_diff.set_xlabel("Birth Weight Difference")
mean_diff.set_ylabel("Proportion per Unit")
mean_diff.set_title("Distribution of Birth Weight Differences");
```



أخيراً، لإيجاد فاصل ثقة بنسبة 95% نوجد النسب المئوية الـ 2.5 و الـ 97.5 في الإحصائية التمهيدية:

```
(np.percentile(mean_differences, 2.5),
 np.percentile(mean_differences, 97.5))
```

```
(-11.36909646997882, -7.181670323140913)
```

فاصل الثقة هذا يسمح لنا بالقول بنسبة ثقة 95% أن الفارق في متوسط الأوزان للمواليد المجتمع الإحصائي بين -11.37 و -7.18 أونصة.

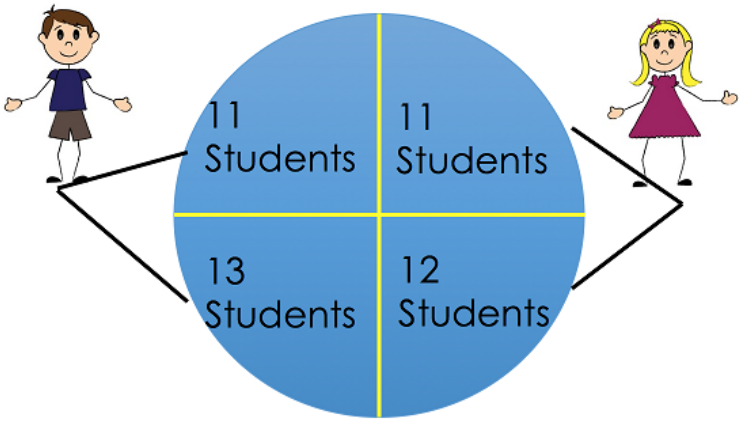
ملخص اختبار الفرضيات و مجال الثقة

في هذا الجزء، راجعنا اختبار الفرضيات باستخدام اختبار التبادل وفاصل الثقة مع التمهيد. للقيام باختبار الفرضية، يجب علينا طرح فرضية العدم والفرضية البديلة، اختيار اختبار الإحصائية المناسب، والقيام بخطوات الاختبار لحساب قيمة الاحتمالية p-value. لإنشاء فاصل ثقة، نختار اختبار إحصاء مناسب، تمهيد العينة الأصلية لإنشاء توزيع تجريبي لاختبار الإحصائية، ونقوم بتحديد الكميات المقابلة لمستوى الثقة الذي لدينا.

اختبار التبادل

هناك عدة حالات نرغب فيها في إجراء اختبار التبادل لتجربة واختبار فرضية والتعرف أكثر على البيانات. اختبار التبادل يعتبر اختبار من النوع الغير معلمي Non-Parametric Test يسمح لنا بإيجاد استنتاجات دون القيام بافتراضات إحصائية والتي تعتبر من نوع الإحصاء المعلمي التقليدي Parametric Test.

أحد الأمثلة الواضحة لاستنتاجات التبادل هو في بيانات تقييم الطلاب للتدريس (Student Evaluation of Teaching (SET بواسطة بورنق، أوتوبوني و ستارك (2016). في هذه الدراسة، 47 طالبًا تم توزيعهم بشكل عشوائي على واحد من أربع أقسام. هناك معلمان يدرسان قسمين؛ إحدى المعلمين ذكر والآخر أنثى. في قسمين، تم التعريف بالمعلمان باستخدام أسمائهم الحقيقية. في الأقسام الأخرى تبادل المعلمان الأسماء.



لم يقابل الطلاب المعلمان وجهاً لوجه. كان التفاعل بين الطلاب والمعلمان ألكترونياً. تم تنسيق إعادة الواجبات إلى الطلاب مع الدرجات والملاحظات في نفس الوقت بين المعلمان. أيضاً لدى المعلمان نفس المستوى من الخبرة. في نهاية الفصل، قيّم الطلاب المعلمان في أدائهم لتصحيح الواجبات وإعادتها لهم. أراد المؤلف أن يتحقق إذا كان للجنس تأثير على تقييم الطلاب للتدريس.

الإعداد للتجربة

أجرينا اختبار الفرضية باستخدام فاصل قطع القيمة الاحتمالية p-value يساوي 0.05.

في **النموذج Model**، لكل مدرس احتمالان للتقييم من كل طالب، واحد لكل جنس متصور لهم (Perceived). لكل طالب فرصة متساوية ليتم تعيينه لتقييم أي من الجنسين في الاختبار (الجنس الحقيقي Gender، الجنس المتصور لهم Perceived Gender) للمعلمان. أخيراً، يقيم الطلاب مدرسيهم بشكل مستقل ومنفصل عن بعضهم.

فرضية العدم Null Hypothesis في هذه التجربة هي الجنس المتصور من الطلاب ليس لدية أي تأثير على تقييم الطلاب للتدريس وأي فرق نلاحظه في التقييم كان بالصدفة. بمعنى آخر، التقييم لكل مدرس يجب ألا يتغير سواء كان يتصور الطلاب أنهم ذكور أو إناث. يعني ذلك أن كل مدرس في الحقيقة سيحصل على تقييم واحد فقط من كل طالب.

الفرضية البديلة Alternative Hypothesis هي أن الجنس المتصور من الطلاب للمدرسين لدية تأثير على تقييم الطلاب للتدريس.

اختبار الإحصائية Test Statistic هو فرق المتوسطات لتقييم الطلاب للمدرسين بتصورهم أنهم ذكور أو إناث. بشكل طبيعي، نتوقع أن تكون النتيجة قريبة من 0 إذا لم يكن للجنس تأثير على التقييم. يمكننا كتابة التالي:

$$\mu_{\text{perceived female}} - \mu_{\text{perceived male}}$$

فيها:

$$\mu_{\text{perceived female}} = \frac{\sum_{j=1}^{n_1} x_{1j} + \sum_{j=1}^{n_3} x_{3j}}{n_1 + n_3}$$
$$\mu_{\text{perceived male}} = \frac{\sum_{j=1}^{n_2} x_{2j} + \sum_{j=1}^{n_4} x_{4j}}{n_2 + n_4}$$

وفيها n_i هي عدد الطلاب في المجموعة i و x_{ij} هي تقييم الطالب رقم j في تلك المجموعة i.

من أجل تحديد ما إذا كان للجنس تأثير على تقييم الطلاب للتدريس، نقوم بإجراء اختبار تبادل لإنشاء توزيع تجريبي لاختبار الإحصاء في فرضية العدم. نقوم بإتباع الخطوات التالية:

- تبديل الجنس المتصور من قبل الطلاب الذين لديهم نفس المعلم. لاحظ أننا نقوم بالخلط من الجزئيين اليسار واليمين من الصورة السابقة.
- حساب الفرق في متوسط تقييم الطلاب في المجموعات التي تم تحديد الجنس فيها كذكر أو أنثى.
- التكرار أكثر من مره لإيجاد توزيع تقريبي للعينات لمتوسط الفرق في التقييم بين المجموعتين.
- استخدام التوزيع التقريبي لتوقع احتمالية إيجاد اختبار إحصاء أكثر تطرفاً مما لاحظناه مسبقاً.

من المهم فهم تبرير اختبار التبادل في هذا المثال. في النموذج العدمي، كل طالب سيعطي مدرسه نفس التقييم أياً كان تصوره للجنس. التعيين العشوائي البسيط يشير أن لكل معلم، جميع تقييماتهم لديها نفس الفرصة أن تظهر أيًا كان تصور الطلاب لهم كذكر أو أنثى. لذا، تبديل الجنس لن يؤثر على التقييم إذا كانت فرضية العدم صحيحة.

البيانات

نبدأ ببيانات الطلاب والجنس التالية. هذه البيانات من مسح إحصائي لـ47 طالب في مسجلين في مادة عبر الإنترنت في إحدى الجامعات الأمريكية:

لتحميل البيانات StudentRatingsData.csv اضغط هنا.

```
student_eval = (
    pd.read_csv('StudentRatingsData.csv')
    .loc[:, ["tagender", "taidgender", "prompt"]]
    .dropna()
    .rename(columns={'tagender': 'actual', 'taidgender': 'perceived'})
)
student_eval[['actual', 'perceived']] = (
    student_eval[['actual', 'perceived']]
    .replace([0, 1], ['female', 'male'])
)

student_eval
```

	actual	perceived	prompt
0	female	male	4
1	female	male	5
2	female	male	5
...	...	...	...
43	male	female	4
44	male	female	2
45	male	female	4

```
43 rows × 3 columns
```

معنى الأعمدة كالتالي:

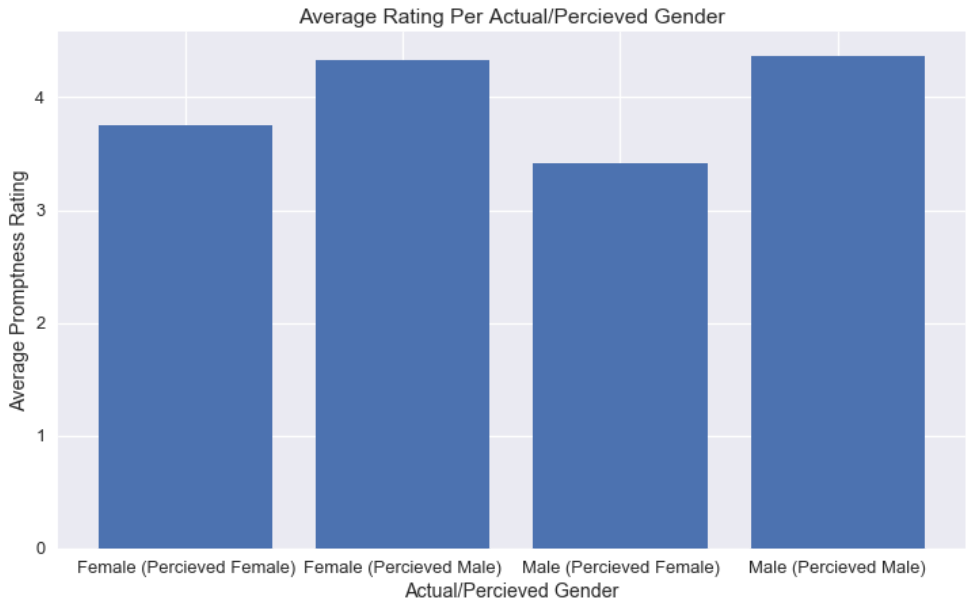
- **actual**: الجنس الحقيقي للمعلم.
- **perceived**: الجنس المعطى (المتصور) للطلاب للمعلم.
- **prompt**: تقييم الطلاب لأداء المعلمين في تصحيح الواجبات وإعادتها لهم من 1 إلى 5.

بعد تحليل ورسم البيانات من التجربة، يبدو أن هناك فرق في مجموعات الطلاب، بتصور الطلاب للمعلم كأنثى حصلت على تقييم أقل من تصور الطلاب للمعلم كذكر؛ ولكن، نحتاج اختبار فرضية أساسي لنرى اذا كان الفرق ببساطة بسبب التعيين العشوائي للطلاب:

```
avg_ratings = (student_eval
 .loc[:, ['actual', 'perceived', 'prompt']]
 .groupby(['actual', 'perceived'])
 .mean()
 .rename(columns={'prompt': 'mean prompt'})
)
avg_ratings
```

		mean prompt
actual	**perceived**	
female	**female**	3.75
	male	4.33
male	**female**	3.42
	male	4.36

```
fig, ax = plt.subplots(figsize=(12, 7))
ind = np.arange(4)
plt.bar(ind, avg_ratings["mean prompt"])
ax.set_xticks(ind)
ax.set_xticklabels(['Female (Percieved Female)', 'Female (Percieved Male)', 'Male (Percieved Female)', "Ma
ax.set_ylabel('Average Promptness Rating')
ax.set_xlabel('Actual/Percieved Gender')
ax.set_title('Average Rating Per Actual/Percieved Gender')
plt.show()
```



تطبيق التجربة

سنقوم بحساب الفرق الواضح في متوسطات التقييم للمعلمين في المجموعات التي تم تحديد جنس المعلم لهم سواء ذكر أو أنثى:

```
def stat(evals):
    # حساب اختبار الإحصاء على الداتا فريم evals
    avgs = evals.groupby('perceived').mean()
    return avgs.loc['female', 'prompt'] - avgs.loc['male', 'prompt']

observed_difference = stat(student_eval)
observed_difference
```

```
-0.79782608695652169
```

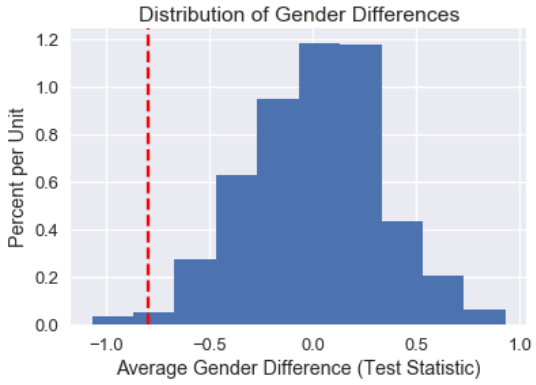
نرى أن الفرق -0.8، في هذه الحالة، متوسط التقييم لمن تم تحديد وإبلاغهم عن جنس المعلم كأنثى اقل بحوالي 1 في المقياس من 1 إلى 5. في هذا المقياس، يظهر أن الفرق كبير جداً. بإجراء اختبار التبادل، يمكن أن نلاحظ فرقاً بهذا الحجم الكبير في نموذج العدم.

الآن، يمكننا تبديل الجنس المصور لكل معلم وحساب اختبار الإحصائية 1000 مره:

```
def shuffle_column(df, col):
    # تنتج لنا الدالة نسخة جديدة من البيانات بعد خلطها
    result = df.copy()
    result[col] = np.random.choice(df[col], size=len(df[col]))
    return result

repetitions = 1000

gender_differences = np.array([
    stat(shuffle_column(student_eval, 'perceived'))
    for _ in range(repetitions)
])
```

نرسم توزيع البيانات التقريبي للفرق في التقييم باستخدام التبادل، أظهرنا القيمة التي سبق أن اطلعنا عليها بالخط الأحمر المتقطع:

```
differences_df = pd.DataFrame()
differences_df["gender_differences"] = gender_differences
gender_hist = differences_df.loc[:, "gender_differences"].hist(density=True)
gender_hist.set_xlabel("Average Gender Difference (Test Statistic)")
gender_hist.set_ylabel("Percent per Unit")
gender_hist.set_title("Distribution of Gender Differences")
plt.axvline(observed_difference, c='r', linestyle='--');
```



من عملية الحساب التالية، نجد أن القيم الـ1000 التي قمنا بمحاكاتها للبيانات، 18 فقط كان الفرق فيها عالي كالتي رأيناها. لذا، قيمة الاحتمالية p-value أقل من قيمة الفصل 0.05 لذا نرفض فرضية العدم ونفضل الفرضية البديلة:

```
# متغيرات التوزيع
sample_sd = np.std(gender_differences)
sample_mean = np.mean(gender_differences)
# حساب القيمة الأعلى يميناً
num_sd_away = (sample_mean - observed_difference)/sample_sd
right_extreme_val = sample_mean + (num_sd_away*sample_sd)
# حساب قيمة الإحتمالية p-value
num_extreme_left = np.count_nonzero(gender_differences <= observed_difference)
num_extreme_right = np.count_nonzero(gender_differences >= right_extreme_val)
empirical_P = (num_extreme_left + num_extreme_right) / repetitions
empirical_P
```

```
0.018
```

ملخص اختبار التبادل

من خلال مراجعة اختبار التبادل، رأينا أن تقييم الطلاب للتدريس منحازة ضد المعلمين الإناث بقيمة كبيرة و هامة إحصائياً.

هناك دراسات أخرى اختبرت أيضاً الانحياز في تقييم التدريس. بناءًا على بورنق، أوتوبوني و ستارك (2016)، هناك اختبارات تبادل أخرى تم إجراءها افترضت أن تقييم المعلمين الذكور والإناث هي عينات عشوائية مستقلة من مجتمع إحصائي موزع بشكل طبيعي مع تباينات متساوية؛ هذا النوع من تصميم التجارب لا يتطابق مع فرضية العدم، مما يجعل قيمة الاحتمالية p-value مضللة.

على العكس، بورنق، أوتوبوني و ستارك (2016) إستخدمو اختبار تبادل بناءاً على توزيع عشوائي للطلاب في الأقسام. تذكر عندما قمنا باختبار التبادل، لم نقم بأي فرضيات عن التوزيع في البيانات. في هذه التجربة، لم نفترض أن الطلاب، تقييم الطلاب للتدريس، الدرجات، أو أي من المتغيرات الأخرى تحتوي على عينات عشوائية من أي مجتمع إحصائي، مجتمع إحصائي أقل بكثير و بتوزيع طبيعي.

عند اختبار فرضية، من المهم اختيار تصميم تجربتك وفرضية العدم من أجل الحصول على نتائج موثوقة.

المصطلحات العربية وترجمتها

في هذه الصفحة سيتم كتابة المصطلحات الإنجليزية المستخدمة في الكتاب ومرادفها بالعربية، في كثير من الأحيان لا يوجد مصطلح عربي مناسب وواضح المعنى لكلمة ما، فسيتم استخدام المصطلح العربي الشائع إستخدامة بالإضافة إلى شرح للمعنى أو تعريف إن وجد، إذا أحتاج الأمر سيتم إضافة روابط باللغة العربية أو الإنجليزية للمساعدة لشرح أكثر عن المصطلح ومعناه بشكل صحيح:

المصطلح الإنجليزي	المصطلح العربي	المعنى	روابط مساعدة
Dataset	البيانات	مجموعة بيانات متكونة على شكل جدول أو قاعدة بيانات	
Data Design	تصميم البيانات	كيفية بناء البيانات، تختلف كل قاعدة بيانات عن أخرى حسب مُصممها	
Series	مجموعة (مصفوفة أحادية البعد)	مجموعة بيانات متكونة على شكل مصفوفة أحادية البعد	
Dictionary	قاموس	مشابهة للمصفوفات لكن تحتوي على مفتاح لكل قيمه Key/Value	
Vector	متّجه	طريقة لعرض البيانات على شكل مصفوفة بسطر واحد أو عمود واحد	
Matrix\Matrices	مصفوفة / مصفوفات	طريقة لعرض البيانات، تستخدم بشكل كثير في العمليات الحسابية وخوارزميات تعلم الآلة وغيرها، تحتوي على أشكال وأحجام مختلفة	

المصطلح الإنجليزي	المصطلح العربي	المعنى	روابط مساعدة
Class	الكلاس/ الصنف	هي جميع الدوال، المتغيرات، الخصائص التي تتعلق بالكائن Object	
Population	المجتمع الإحصائي	مجموعة من البيانات لديها عناصر متشابهه	
Quota Sampling	عينة الحصة	عينة تحتوي على بعض أو كل صفات و سمات المجتمع الإحصائي	
Parameter	معْلَمة	في الإحصاء: مقياس إحصائي يحسب من كافة مفردات المجتمع دون استثناء، في البرمجة: قيمة يتم تمريرها للدوال أو العمليات الحسابية	
Variable	متغير	في الإحصاء، قيمة متغيرة من شخص لآخر، مثل الطول والوزن	
Random variable	متغير عشوائي	متغير ليس له قيمة ثابتة	
Sample	عيّنة	جزء من المجتمع الإحصائي يستخدم لتمثيله	
Classifier	مُصنف	الخوارزمية التي تقوم بتصنيف البيانات وتقسيمها	
Model	نموذج	يطلق على النتيجة بعد تعلم الآلة، النموذج يتعلم عن طريق الخوارزميات، ثم يتم أختبارة على بيانات حقيقية	
Features	خصائص/ متغيرات	عند تكوين النموذج، نحدد الخصائص التي يعتمد عليها النموذج لإجراء التنبؤات	
Fit	ضبط	ضبط النموذج بأفضل المتغيرات لتجهيزه للتدريب	
Train	تدريب	تدريب النموذج على البيانات المتوفرة ونتائجها	
Predict	التوقع / التنبؤ	النتائج من النموذج بعد تدريبه	
Weight	الوزن	متغير قابل للتعلم، يستخدم أرقام عشوائية داخل النموذج حتى يصل لنتيجة مناسبة	
Noise	التشويش	بيانات عشوائية وغير منتظمة مع بقية البيانات	
Bias	تحيز / انحياز	انحياز البيانات لجزء أو قيمه معينة، مدى بعد النتائج من الأجوبة أو التوقعات الصحيحة	
Variance	التباين	عندما يؤدي النموذج بشكل جيد فقط على بياناته، عندما يواجه بيانات جديده، لا يؤدي بنفس الجودة التي يؤديها على بيانات التدريب	
Bootstrap	بوتستراب	عينة من المجتمع تمثله بشكل جيد	
Gradient Descent	نزول اشتقاقي	طريقة لتحسين النتائج، في كل خطوة يتم تكرار العمليات حتى تصل للمتغيرات المثالية والتي تقلل من دالة التكلفة	
Cost Function	دالة التكلفة	دالة تقيس مدى سوء نتائج النموذج	
Loss Function	دالة الخسارة	دالة تستخدم لقياس النتائج التي تم تصنيفها بشكل خاطئ عن طريق حساب مدى بعدها عن النتائج الحقيقية	
Empirical Risk	الخطر التجريبي	دالة نتيجتها تأتي عن طريق إيجاد متوسط دالة الخسارة المستخدمة في البيانات	
Activation function	دالة التنشيط	توجد أنواع مختلفة من دوال التنشيط، جميعها هدفها استقبال مدخلات في الشبكات العصبية وإخراجها بعد التعلم منها	
Regularization	الضبط	طريقة للتقليل من تباين النموذج والتخلص من فرط التخصيص	
Normalization	التسوية	إعادة تعين القيم في الأعمدة الرقمية لتكون بين الرقمين 0 و 1 بشكل مدرج دون التأثير على الفرق بين تلك القيم	
Standardization	التوحيد	طريقة أخرى لإعادة تعين القيم في الاعمدة الرقمية لتكون أقرب إلى المتوسط	
Supervised learning	التعلم الموجّه	يعتمد على البيانات المُعلّمة في التعلم، يتم تقديم هذه البيانات ونتائجها وتدريب الآلة عليها	
Unsupervised learning	التعلم غير الموجّه	عكس التعلم الموجّه، الآلة تُعطى بيانات وتتعلم من نفسها عن طريق محاولة إيجاد أنماط، يتم ذلك عن طريق تجميع البيانات التي لدينا إلى مجموعات حسب ما تتشابه فيه	
Confusion Matrix	مصفوفة الدقة	أداة تستخدم لقياس أداء ودقة نتائج النماذج	

المكاتب المستخدمة في الكتاب ودوالها

مقدمة

هذا الملحق يحتوي على جداول لدوال المكتبات المستخدمة في هذا الكتاب `matplotlib`، `seaborn`، `pandas` و `scikit-learn`. هدف هذه الجداول أن تكون ملخص ومرجع مفيد للدوال الأكثر استخداماً في هذا الكتاب.

لكل مكتبة، قمنا بإضافة الدالة، الفصل الذي استخدمت في الدالة أول مرة، وشرح بسيط عن طريقة عملها.

Pandas

الدالة	الفصل	الوصف
`pd.DataFrame(data`	البيانات المجدولة ومكتبة بانداز	إنشاء DataFrame من مصفوفة أو قاموس ثنائي الأبعاد `data`
`pd.read_csv(filepath`	البيانات المجدولة ومكتبة بانداز	قراءة ملف من النوع CSV في المسار `filepath` كـ DataFrame
`pd.DataFrame.head(n=5` `pd.Series.head(n=5`	البيانات المجدولة ومكتبة بانداز	إظهار أول `n` أسطر في مصفوفة أحادية البعد أو DataFrame
`pd.DataFrame.index` `pd.DataFrame.columns`	البيانات المجدولة ومكتبة بانداز	إظهار الأرقام التسلسلية للDataFrame و أسماء الأعمدة
`pd.DataFrame.describe()` `pd.Series.describe()`	التحليل الاستكشافي للبيانات	عرض معلومات إحصائية عن مصفوفة أحادية البعد أو الDataFrame
`pd.Series.unique()`	التحليل الاستكشافي للبيانات	عرض القيم الفريدة (الغير مكررة) في مصفوفة أحادية البعد
`pd.Series.value_counts()`	التحليل الاستكشافي للبيانات	عرض عدد مرات تكرار كل قيمة فريدة (غير مكررة) تظهر في المصفوفة أحادية البعد
`df[col]`	البيانات المجدولة ومكتبة بانداز	من الـ `df` DataFrame، أظهر العامود `col` كمصفوفة أحادية البعد
`df[[col]]`	البيانات المجدولة ومكتبة بانداز	من الـ `df` DataFrame، أظهر العامود `col` كDataFrame
`df.loc[row, col]`	البيانات المجدولة ومكتبة بانداز	من الـ `df` DataFrame، أظهر لنا السطر ذو الاسم `row` واسم عموده `col`؛ يمكن تبديل `row` بمصفوفة أحادية البعد بقيم منطقية (True/False)
`df.iloc[row, col]`	البيانات المجدولة ومكتبة بانداز	من الـ `df` DataFrame، أظهر لنا السطر ذو الرقم التسلسلي `row` واسم عموده `col`؛ يمكن تبديل `row` بمصفوفة أحادية البعد بقيم منطقية (True/False)
`pd.DataFrame.isnull()` `pd.Series.isnull()`	تنظيف البيانات	إظهار القيم المفقودة في مصفوفة أحادية البعد أو DataFrame
`pd.DataFrame.fillna(value` `pd.Series.fillna(value`	تنظيف البيانات	تعبئة القيم المفقودة في المصفوفة أحادية البعد أو الDataFrame بالقيمة في المتغير `value`
`pd.DataFrame.dropna(axis` `pd.Series.dropna()`	تنظيف البيانات	حذف الأسطر أو الأعمدة
`pd.DataFrame.drop(labels, axis`	تنظيف البيانات	حذف الأسطر أو الأعمدة التي اسمها `labels` من الDataFrame من المحور `axis`
`pd.DataFrame.rename()`	تنظيف البيانات	إعادة تسمية الأسطر أو الأعمدة في DataFrame
`pd.DataFrame.replace(to_replace,` `value`	تنظيف البيانات	تبديل القيمة `to_replace` بالقيمة `value` في الDataFrame
`pd.DataFrame.reset_index(drop=False`	تنظيف البيانات	إعادة تعيين الأرقام التسلسلية في الDataFrame؛ تلقائيًا، يتم حفظ الأرقام السابقة في عمود جديد إلا إذا تم وضع المتغير `drop=True`
`pd.DataFrame.sort_values(by,` `ascending=True`	البيانات المجدولة ومكتبة بانداز	ترتيب الDataFrame بناءًا على العمود في المتغير `by`، الترتيب التلقائي تصاعدي
`pd.DataFrame.groupby(by`	البيانات المجدولة ومكتبة بانداز	إنشاء كائن GroupBy يحتوي على DataFrame مُجَمعة بناءًا على العامود في المتغير `by`
`GroupBy.<function>`	البيانات المجدولة ومكتبة بانداز	تطبيق الدالة `<function>` لكل مجموعة في الكائن GroupBy ؛ مثال: `count()`، `mean()`
`pd.Series.<function>`	البيانات المجدولة ومكتبة بانداز	تطبيق الدالة `<function>` على مصفوفة أرقام أحادية البعد؛ مثال: `median`, `max()`، `mean()`
`pd.Series.str.<function>`	البيانات المجدولة ومكتبة بانداز	تطبيق الدالة `<function>` على مصفوفة نصية أحادية البعد؛ مثال: `split`, `lower()`، `len()`
`pd.Series.dt.<property>`	البيانات المجدولة ومكتبة بانداز	استخراج القيمة `<property>` من مصفوفة تاريخ ووقت أحادية البعد؛ مثال: `date`, `month`، `year`

الدالة	الفصل	الوصف
`pd.get_dummies(columns, (drop_first=False`	—	تحويل القيم الاسمية في العمود `columns` لقيم وهمية Dummy؛ الوضع التلقائي بإبقاء جميع القيم إلا إذا تم تحديد المتغير `drop_first=True`
`(pd.merge(left, right, how, on`	التحليل الاستكشافي للبيانات / قواعد البيانات العلائقية و SQL	جمع اثنين DataFrame في المتغيرات `left` و `right` معاً باستخدام العمود في المتغير `on`؛ بناءًا على طريقة الجمع في المتغير `how`
`(pd.read_sql(sql, con`	قواعد البيانات العلائقية و SQL	قراءة أمر SQL في المتغير `sql` في قاعدة البيانات المتصل بها في `con`، تعود النتيجة كDataFrame

Seaborn

الدالة	الفصل	الوصف
`sns.lmplot(x, y, data, (fit_reg=True`	تصوير البيانات	إنشاء مخطط تشتت لـ x و y من الـDataFrame `data`، وبشكل تلقائي يظهر لنا خط انحدار المربعات الصغرى
`sns.distplot(a, (kde=True`	تصوير البيانات	إنشاء مخطط المدرج التكراري لـ a، وبشكل تلقائي يظهر لنا تقدير للكثافة
`sns.barplot(x, y, (hue=None, data, ci=95`	تصوير البيانات	إنشاء مخطط شريطي لـ x و y من الـDataFrame `data`، يتم تحليلها اختيارياً بالخيار في المتغير `hue`، وبشكل تلقائي رسم خط ثقة بنسبة 95% (والذي يمكن عدم رسمه بتحديد الخيار `ci=None`)
`sns.countplot(x, (hue=None, data`	تصوير البيانات	إنشاء مخطط شريطي لمجموع القيم الفريدة في المتغير x في الـDataFrame `data`، يتم تحليلها اختيارياً بناءًا على المتغير النوعي `hue`
`sns.boxplot(x=None, y, (data`	تصوير البيانات	إنشاء مخطط صندوق لـ y، يتم تحليله اختيارياً بناءًا على المتغير النوعي x، من الـDataFrame `data`
`(sns.kdeplot(x, y=None`	تصوير البيانات	إذا كانت `y=None`، أنشأ مخطط كثافة بمتغير واحد x؛ إذا تم تحديد y، أنشأ مخطط كثافة ثنائي
`sns.jointplot(x, y, (data`	تصوير البيانات	جمع مخطط التشتت الثنائي المكون من x و y من الـDataFrame `data`، مع مخطط كثافة ذو المتغير الواحد يتم رسم خط كل متغير في المحاور
`sns.violinplot(x=None, (y, data`	تصوير البيانات	جمع وأنشاء مخطط الصندوق و مخطط كثافة للمتغير y، يتم تحليله اختيارياً بناءًا على المتغير النوعي x، من الـDataFrame `data`

matplotlib

أنواع الرسومات

الدالة	الفصل	الوصف
`(plt.scatter(x, y`	تصوير البيانات	إنشاء مخطط تشتت لـ x و y
`(plt.plot(x, y`	تصوير البيانات	إنشاء مخطط خطي لـ x و y
`plt.hist(x, (bins=None`	تصوير البيانات	إنشاء مخطط المدرج التكراري لـ x. المتغير `bins` يمكن أن يكون رقم أو أرقام تسلسلية
`(plt.bar(x, height`	تصوير البيانات	إنشاء مخطط شريطي. x تحدد تنسيق الأشرطة، و `height` تحدد طول الأشرطة
`(plt.axvline(x=0`	تصوير البيانات	إنشاء خط عمودي على القيمة المحددة في المتغير x
`(plt.axhline(y=0`	تصوير البيانات	إنشاء خط أفقي على القيمة المحددة في المتغير x

إضافة الرسومات

الدالة	الفصل	الوصف
`matplotlib inline%`	تصوير البيانات	السماح بالرسوم البيانية أن تظهر في نفس الصفحة بدلاً من صفحة منفصلة
`plt.figure(figsize=(3, ((5`	تصوير البيانات	إنشاء رسم بياني بعرض 3 وطول 5 إنش
`(plt.xlim(xmin, xmax`	تصوير البيانات	تحديد حد للمحور x-limits

الدالة	الفصل	الوصف
`plt.xlabel(label)`	تصوير البيانات	تحديد عنوان للمحور
`plt.title(label)`	تصوير البيانات	تحديد عنوان للمحاور / الرسم البياني
`plt.legend(x, height)`	تصوير البيانات	إضافة عنوان تفصيلي / تفسيري للرسم البياني
`fig, ax = plt.subplots()`	تصوير البيانات	إنشاء رسم بياني ومجموعة من الرسوم البيانية داخلة
`plt.show()`	تصوير البيانات	عرض الرسم البياني

scikit-learn

النماذج واختيارها

الإستدعاء	الدالة	الفصل	الوصف
`sklearn.model_selection`	`train_test_split(*arrays, test_size=0.2)`	النماذج والتوقعات	ينتج عنها قسمين من المصفوفة المرسولة في الدالة، نصف فيه 0.8 من المصفوفة والنصف الآخر 0.2
`sklearn.linear_model`	`LinearRegression()`	النماذج والتوقعات	إنشاء نموذج الانحدار الخطي للمربعات الصغرى
`sklearn.linear_model`	`LassoCV()`	النماذج والتوقعات	إنشاء نموذج خطي Lasso (الضبط L1) يختار أفضل المتغيرات باستخدام التحقق المتقاطع
`sklearn.linear_model`	`RidgeCV()`	النماذج والتوقعات	إنشاء نموذج خطي Ridge (الضبط L2) يختار أفضل المتغيرات باستخدام التحقق المتقاطع
`sklearn.linear_model`	`ElasticNetCV()`	النماذج والتوقعات	إنشاء نموذج خطي ElasticNet (الضبط L1 و L2) يختار أفضل المتغيرات باستخدام التحقق المتقاطع
`sklearn.linear_model`	`LogisticRegression()`	النماذج والتوقعات	إنشاء نموذج انحدار لوجستي
`sklearn.linear_model`	`LogisticRegressionCV()`	النماذج والتوقعات	إنشاء نموذج انحدار لوجستي يختار أفضل المتغيرات باستخدام التحقق المتقاطع

العمل مع النماذج

الدالة	الفصل	الوصف
`model.fit(X, y)`	النماذج والتوقعات	ضبط النموذج باستخدام الخصائص X والنتائج y
`model.predict(X)`	النماذج والتوقعات	إيجاد التوقعات لـ X باستخدام النموذج `model`
`model.score(X, y)`	النماذج والتوقعات	إيجاد دقة النتائج للتوقعات القيم X مقارنة بالنتائج الصحيحة في y

إنتهى